**Pekka Mäkiaho and Timo Poranen (eds.)**

# Software Projects 2011-2012


UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 9

TAMPERE 2012

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 9
APRIL 2012

**Pekka Mäkiaho and Timo Poranen (eds.)**

# Software Projects 2011-2012

SCHOOL OF INFORMATION SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 978-951-44-8809-2

ISSN-L 1799-8158
ISSN 1799-8158

# Preface

This report contains project stories of 14 software development projects from academic year 2011-2012. The students came from Project Work and Software Project Management courses. The stories describe what kind of experiences groups got during their project and what was the software product that came out from the project. In the end of each story there are project statistics.

**Table 1: General project statistics.**

| Project | Type | Client | Dev. Mod. | Group | Hours |
|---|---|---|---|---|---|
| Navilendar | Mobile | Other | Lean | 3+5 | 1632 |
| Visual Money | WWW | Company | Scrum | 2+3 | 1144 |
| Ylioppilasteatteri | WWW | Assoc. | Scrum | 2+5 | 1188 |
| Asiaa! | WWW | SIS | Scrum | 2+4 | 972 |
| Editaattori | WWW | University | Scrum | 3+5 | 1300 |
| Shakkilinna | Mobile | Company | Scrum but | 2+5 | 1312 |
| Math.fi | WWW | Other | Scrum | 2+5 | 1200 |
| Retu | WWW | Other | Scrum | 2+5 | 1350 |
| Moodle | WWW | SIS | Scrum | 2+5 | 1033 |
| ComputerVisionGame | Appl. | SIS | Scrum | 2+5 | 1594 |
| Snakes and Ladders | Mobile | Company | Agile | 2+5 | 1520 |
| CpVis | WWW | Demola | Scrum but | 2+5 | 1662 |
| Category Browser | WWW | Demola | Feature driven | 2+5 | 1502 |
| Cows | WWW | Demola | Incremental | 2+5 | 1225 |

Table 1 gives an overview of projects, project type (WWW, mobile or standalone application), client (SIS=School of Information Sciences, Assoc.= non-commercial associations, Company, Demola or Other non-commercial client), development model, group size (number of managers + number of developers) and working hours of the project. Project types which are marked with an asterisk (*), developed further an existing software.

Although ten projects applied Scrum development model, they had one major difference when compared to standard Scrum: daily scrum meetings were mainly organized virtually using IRC or similar real-time messaging systems, or the daily meetings were omitted.

Table 2 contains general course statistics (number of projects and usability teams, number of students in the courses and average project size in working hours) starting from year 2005.

**Table 2: Course statistics 2005-2012**

| Academic year | Projects | Usability teams | PW students | SPM students | Average project size |
|---|---|---|---|---|---|
| 2005-6 | 19 | 1 | 98 | 8 | 1008 |
| 2006-7 | 18 | 2 | 87 | 34 | 1089 |
| 2007-8 | 14 | 1 | 70 | 29 | 997 |
| 2008-9 | 10 | 1 | 60 | 39 | 1643 |
| 2009-10 | 15 | 1 | 80 | 34 | 1151 |
| 2010-11 | 13 | 1 | 70 | 27 | 1230 |
| 2011-12 | 14 | 0 | 67 | 30 | 1331 |

School of Information Sciences offered version control services (subversion) and Redmine-project management tool for all projects. During the course we also used projectWiki for maintaining course and project related documentation: https://projectwiki.cs.uta.fi. The wiki also contains some articles on project management and project management tools, including lists of end-products currently in use, course related publications and course related videos:

- https://projectwiki.cs.uta.fi/wiki/Finished_projects
- https://projectwiki.cs.uta.fi/wiki/Course_publications
- https://projectwiki.cs.uta.fi/wiki/List_of_project_videos

Course staff thanks our clients and students for great projects!

Pekka Mäkiaho and Timo Poranen

Tampere, April 2012

# Navilendar

## Overview

Navilendar is a mobile application that unites the essential features of your calendar and map to a single application. By combining their features it will create new ways to display your calendar appointments and add more functionality to them.

With calendar events' locations it is possible to pinpoint and visualize each appointment or event using a map. Once displayed on the map it can even plot a route between two appointments. This way Navilendar makes sure you reach your regular and irregular daily appointment in time.

Original plan was to implement features which make it possible for the user to use Navilendar to make automatic or semiautomatic notifications to event participants in case user cannot make it to an appointment on time. These are just the tip of the iceberg of features planned for implementation.

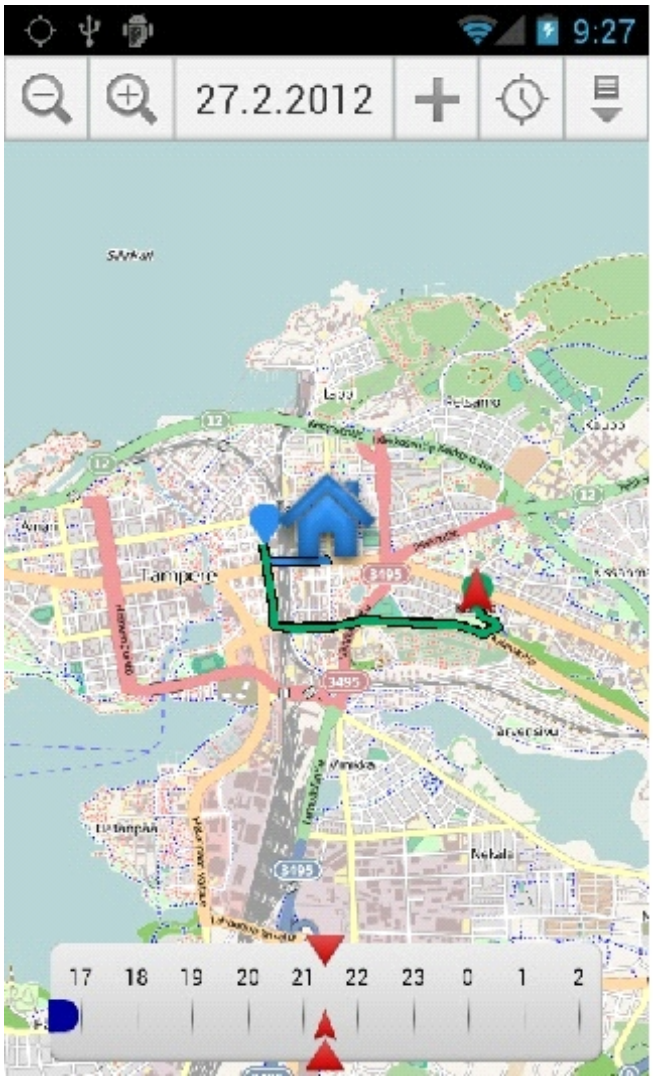



*Image 1: Vertical view.*

In image 1 you can see two events and routes between them of the chosen date. By scrolling the bar at bottom called Timebar you can see where you should be at the given time for example if you would have to be moving to an event at that time, you will see the position you should be at. The two triangles at top and bottom of Timebar show what time you are viewing at the moment and the red triangle shows the current time. The map we are using is OpenStreetMaps and to get it to work on Android we used osmdroid. We had to use osmdroid because it was against the developer license to use Google Maps to draw the routes.

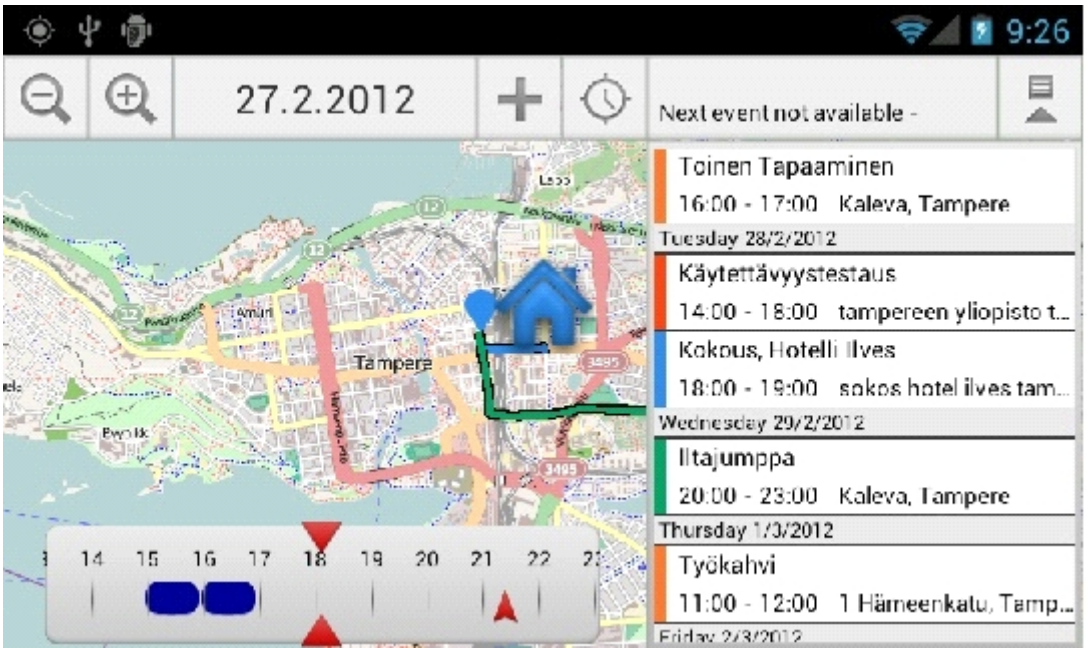



*Image 2: Horizontal view with event drawer opened.*

In image 2 you can see the vertical view and the event drawer. Event Drawer contains the events from your calendar. By clicking on an event you move to that particular event on the map and can view the route to there.

## Organisation and management

Our group was split into two teams at start. The teams were Code team which handled the backend and logic side of Navilendar and the UI team which handled the designing of the user interfaces and testing the product at the end.

We had three managers out of whom Matti Nieminen was the administrative manager in charge of common tasks, Toni Järviluoto was in charge of the Code team and Erkki Heikkonen was in charge of the UI team. Our organization can be best viewed in the next chart.

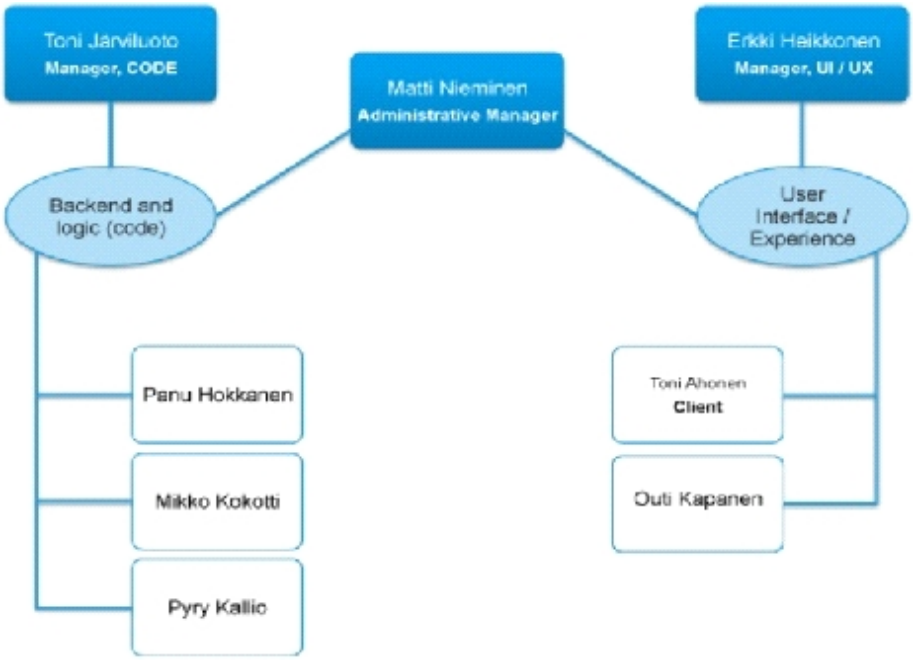



*Diagram 1: organizational chart.*

*Image 3: Our group from left to right: Toni Ahonen, Outi Kapanen, Erkki Heikkonen, Panu Hokkanen, Toni Järviluoto, Mikko Kokotti, Matti Nieminen and Pyry Kallio.*

**Administrative Manager Matti Nieminen**

I worked as "organizational" project manager. I tried to keep everyone in touch with each other, make sure that deadlines were met, reserved meeting rooms and organized weekly meetings and reviews. I was also responsible for weekly reports.

**UI team Manager Erkki Heikkonen**

My main responsibility was managing the usability and user experience team. I organized and managed weekly UI-meetings in which we planned and designed the product. In order to enhance the team productivity I also organized Photoshop trainings for the team. My team was responsible for planning and executing usability testing.

**Customer / UI Team member Toni Ahonen**

I worked in graphic design and developed existing properties further. I helped to create part of the user interface and performed assignments given by the manager. As a customer, I had close contact with the development team and we were able to develop ideas quickly.

**UI Team member Outi Kapanen**

My main job was to plan and design the product with the UI Team. I helped to create the user interface and mainly did the assignments given by the manager. We also planned and executed usability testing.

**Code team Manager Toni Järviluoto**

I was the manager of the Code team. My main responsibility was to track the code team's progress and make sure everybody of the Code team has something to do. I also forwarded possible questions towards UI team. I organized code workshops which were one to six hour long sessions and two weekend long code camps for the Code team when needed.

**Code team member Panu Hokkanen**

My main responsibility in the project was to include calendar in our application. It was my job to get calendar to sync with our application so we could read, write and modify data. I was also responsible for implementing ui designs with our application. I attended many weekly UI-meetings to serve as a link between the UI Team and the Code team. I also hosted the code camps at my place!

**Code team member Pyry Kallio**

My main responsibility area in the project was the implementation of map related features. This included the map view on the background of main view, drawing routes and locations of calendar events and updating and showing user's location using GPS.

**Code team member Mikko Kokotti**

For the whole course, I've been programming a custom component called timebar to the application. It has taken a lot of time to learn to build a custom component, planning and programming. At the end of the course, I've been integrating the component with other components in the application with help of the rest of the Code team.

## Methods and tools

| Name | Version | Purpose | Usefulness |
|---|---|---|---|
| Eclipse IDE | Indigo | Main development | Extremely useful |
| Android SDK | r13-16 (2.3-4.0) | Tools needed to compile and develop Android code | Extremely useful |
| Subversion | | Version control | Extremely useful |
| Redmine | 1.3.1 | Project information sharing, e.g. features, meeting minutes, etc. | Very useful |
| Photoshop | Photoshop | Creating sketches and UI elements | Very useful |
| Google Docs | - | Document creation | Very useful |
| Kanbanery | - | Tracking tasks and visualizing the development process | Not useful, was dropped |

*Table 1: Methods and tools.*

## Project phases and development model

In the beginning of the project the managers agreed on applying Kanban in our development process. Kanban is mainly a tool used to direct workflow an it can be used in various Agile methods such as Scrum or Lean Software Management. Out of those options our method was close to Lean Software Management. We continuously directed resources and efforts in developing features that provided actual value from the eyes of the customer. This was done on weekly basis.

We had quite much challenges in finding a suitable virtual Kanban tool to use. In the beginning of the project we implemented Redmine to handle our information distribution and progress tracking. We requested the Redmine Kanban plug-in but that was not ready to be used and it had some technical difficulties so it failed miserably. Later we implemented a third-party service called Kanbanery, but at that point we had gotten very used to Redmine. Using two tools didn't seem very good so the Kanbanery usage was not pursued very actively. After a while it was totally dropped out. So in the end our development model boiled down to Redmine tracking, active co-operation within the development team and pull system for features that was directed by the customer.

We had various workshops during the project. These workshops were organized on a need and team basis. UI team had workshops for Photoshop usage and getting used to version control using SVN and time management with Redmine. Code team also held a number of shorter workshops in addition to the two LAN events which we called Code Camps where they took great leaps in development.

Virtual communications were done using email, IRC and bulletin boards. Email was

used to distribute information that was important and need to be stored and to communicate between the Code and UI team. IRC was used by code team for development purposes and daily communication. Bulletin board was used for UI teams information sharing and planning activities.

Our project's phases for the code team were mostly practicing, implementation, integration and testing. At the beginning of two of our biggest phases implementation and integration we held the Code Camps. On the first Code Camp we made the initial project and started implementing the main features. On the second Code Camp we started integrating the different units together. Implementation and integration took so long that we did not have time for testing but we managed to achieve the minimum goals plus more of our project.

| **Meeting** | **Content** | **Date** | **Course deadline** |
|---|---|---|---|
| Kick-off meeting | First meeting of the team | 19.9.2011 | - |
| Preliminary Analysis Meeting | Review of the Preliminary Analysis | 22.9.2011 | 23.0.2011 |
| Project Plan Inspection | Review of the Project plan with Pekka Mäkiaho | 6.10.2011 | 7.10.2011 |
| Review I (Checkpoint I) | Review of the updated project plan and project progress Pekka Mäkiaho | 28.11.2011 | 28.11.2011 |
| Review II (Checkpoint II) | Review of the project progress with Pekka Mäkiaho | 13.1.2012 | 16.12.2011 |
| Review III (Checkpoint III) | Review of the project progress with Pekka Mäkiaho | 3.2.2012 | 31.1.2012 |
| Final Report Meeting | Review of the Final Report with Pekka Mäkiaho | 9.3.2012 | 16.3.2012 |

*Table 2: Important checkpoints and meetings.*

## Experiences

### Foreseen risks

| Risk | Analysis |
|---|---|
| Technology problems /<br>Tools and skills | Countermeasure<br>Adequate time to let developers introduce themselves to Android. Possibly organise Android workshops.<br>Analysis<br>All of the members learned necessary skills. We mainly got problem with tools and technology but more on that on 4.2 risks not forseen. |
| Quitting team members | Countermeasure<br>First and foremost good commenting of code and clear documentation to enable easier work transfers. Motivation through changing tasks and lightening the work load<br>Analysis<br>We did not have any members quitting, but as hours started to accumulate and other courses were in progress there was some loss of interest towards the end. |
| Working and studying during project | Countermeasure<br>Adjust the personal schedules depending on availability and try to reach a trade-off between productivity and flexibility.<br>Analysis<br>This went fairly good, except on the part mentioned on previous subject. |
| Documentation problems | Countermeasure<br>Try not to overburden developers with excessive documentation requirements.<br>Analysis<br>Documenting has mainly been a job for the managers, but members have done their part when needed. Documenting and commenting the code could have been done better because as of the moment it seems the |

| | |
|---|---|
| | project will be continued. |
| Lack of testing | Countermeasure<br>Keeping statistics about lines of code or modules that have not been tested. Not accepting new major features to trunk without testing the old ones first. Using test coverage tools when possible.<br>Analysis<br>Some features were tested, but time given to testing was cut mostly because of time constraints. We would have had more time for testing if we could have used Google's maps and calendar at the first place. Also it would have been extremely difficult to write tests for some of the features. |
| Android emulator performance issues | Countermeasure<br>Find alternative ways of testing during development. Possibly need to acquire more test devices.<br>Analysis<br>The emulator worked in various degrees depending on the computer. With some it worked well enough but on some computers it wasn't running smoothly enough. We also received information that the emulator can't be used to test calendar features but near the end of the project we managed to get the emulator to work. |

*Table 3: Foreseen risks.*

**Unforeseen risks**

| **Risk** | **Analysis** |
|---|---|
| Android not being as open as we thought it is | Android wasn't as open as we thought it is. We had to abandon using default Android map because of it's limited possibilities with drawing the route. We also had problems with the calendar, because Android didn't have any calendar API in current versions so we decided to change the version for which we were developing to to Android 4.0 which created the problem that we had to wait until it was released. |
| Different team members working different hours | Even though the total amount of working hours (2208 hours) was really great starting |

| | |
|---|---|
| | point, not all members worked the same amount. Some team members "capped" their hours few weeks before the course was ending and other had the bare minimum. It was problematic to find a member for some tasks near the end of the course.<br><br>Also other team members liked to work from Monday to Friday and others during weekend. This meant that in weekly meetings other members had finished all the tasks for the week and others had not even started. |
| Kanban not working in our project | Trying Kanban without physical board and not meeting more often than once a week was a mistake. The already implemented Redmine Issue tracking provided a way to track features so eventually the virtual Kanbanery board was forgotten and the team utilized a custom process, which proved to be working in our environment. |
| Team members getting work | During the project some members in our team got jobs and some already had jobs. This caused problems when scheduling meetings. Also some time that team members thought they had could not be used to develop Navilendar due to surprising shifts in work. |
| Computer trouble | During the course Outi's computer broke and it consumed a lot of time to get her a new one. Also Toni Järviluoto has had some computer issues needing to reinstall his laptop three to four times during the course. |

*Table 4: Unforeseen risks.*

## *Statistics*

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 3+5 | Lean | 19.9.2011 | 16.3.2012 | 180 | 1631.77 |

*Table 1: General project information.*

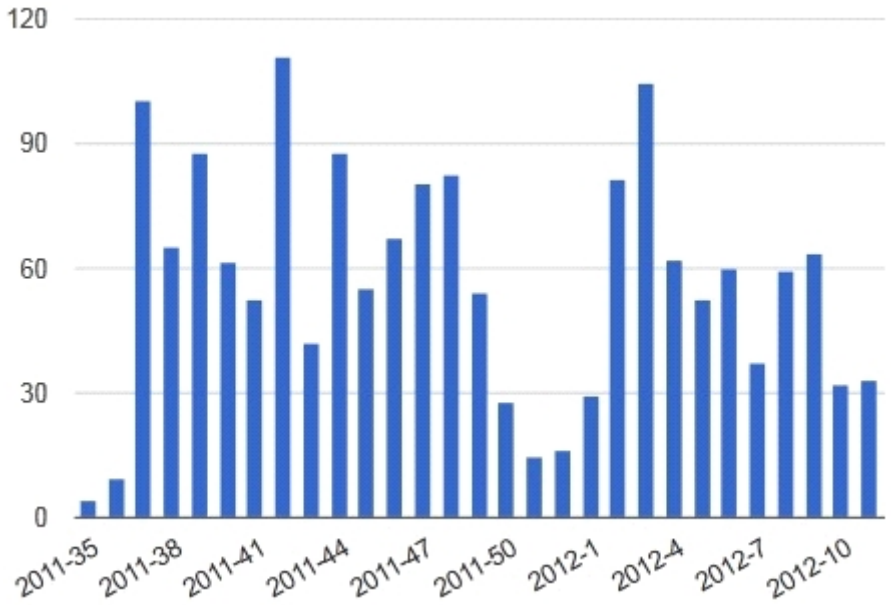



*Diagram 2: Working hours by week.*

| Activity | Planning and management | Req. specification. | Design | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 724.51 | 17.35 | 130.25 | 366.83 | 23.65 | 39.45 | 24.50 | 232.93 | 50.05 | 1609.52 |
| % | 45% | 1% | 8% | 23% | 1% | 2% | 2% | 15% | 3% | 100% |

*Table 2: Group effort by activity.*

| Number of implemented requirements | Number of rejected requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|---|
| 20 | 4 | | 5 | 3 | - | - |

*Table 3: Requirements and high-level design outcomes.*

| Language | JAVA, XML |
|---|---|
| LOC | 8355 |
| SLOC | 6445 |
| Classes | 44 |
| Functions | 285 |
| Code revisions | 68 |

*Table 4: Codelines*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 10 | 1 |
| Project Plan | 25 | 6 |
| UI quick guide | 1 | 1 |
| Final report | 35 | 1 |
| Project's story | 12 | 2 |
| Weekly reports | 27 | |
| Usability test plan | 7 | 3 |
| Usability test script | 6 | 1 |
| Usability heuristic review | 5 | 3 |

*Table 5: Documents.*

.

# Visual Money

## Overview

VisualMoney visualizes any organization's budget in graphical presentation. It's very simple. User inputs the amount of money he pays of membership fee or taxes to organization and as the result program displays that amount assigned in different categories. The data used in visualization could be for example balance sheet or financial statement. There are three different visualization types to present same information.

.../TOAS_2011_Tilinpaatos/Sosiaali- ja terveys

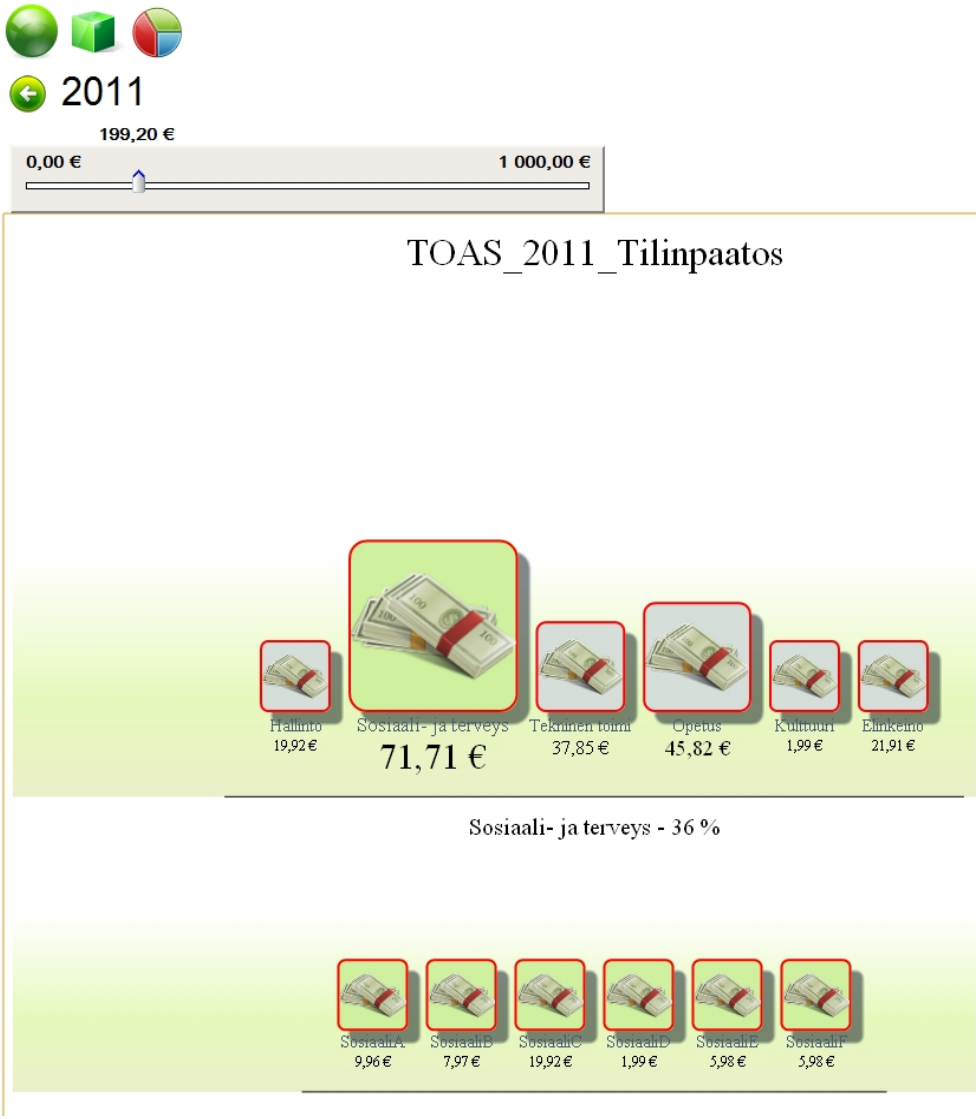



*Illustration 1: An example visualization.*

# 1. Organization and management

Project managers
- Pekka Ihalainen
- Antti Kantola

Project workers
- Erno Rantanen, programming
- Kalle Myllymaa, programming
- Juho Eskola, database

Two project workers quit the course.

# 2. Methods and tools

Eclipse IDE

Eclipse was our ide of choice to develop the code. It was fairly easy to integrate it in version management server. It also had helpful version management tools.

SVN ( sis.uta.fi )

Must have tool for effective team work. We experienced no problems.

Virtual Server( visualmoney.sis.uta.fi )

Virtual server was fresh install of some linux distribution. We got root access for ourselves. It was nice.

Redmine( wiki etc )

Redmine seemed good at start but as the information in it kept growing it came hard to maintain it. So we didn't. The repository browser was nice and hour reporting.

Skype

We used this tool for instant messaging. It was good. You could see the message log for the time you were offline. And there were audio and video call options which we used.

# 3. Project phases and development model

Our development model was scrum. We didn't have daily meetings but we kept our Skype channel active all times.

14.9.2011 First meeting with the project team.

16.9.2011 First meeting with the client.

22.9.2011 Scrum sprints start.

23.9.2011 Requirements specifications.

28.9.2011 Preliminary analysis returned.

29.9.2011 Preliminary analysis review.

6.10.2011 Project plan returned.

7.10.2011 Project plan review

7.11.2011 Personal report I

1.12.2011 Midterm presentation

15.1.2012 Personal report II

14.3.2012 Final presentation

## 4. Experiences

We had two dropouts in our team so we clearly had some motivation issues. We also had problems arranging meetings. It was hard to find common time for meeting. Also one memeber lived in distance. We also had trouble with requirements management.

We should have kept the group more tight including customer. That meaning more meetings. The workshops we didn't try.

## 5. Statistics

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+3 | Scrum | 14.9.2012 | 13.4.2012 | 212 | 1143.50 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 260 | 23,25 | 61 | 486 | 45,50 | 10 | 9 | 116,5 | 132 | 1143.50 |
| % | 22,76 | 2,03 | 5,33 | 42,50 | 3,98 | 0,87 | 0,79 | 10,19 | 11,54 | 100% |

*Table 2: Group effort by activity.*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 106 | 14 | 0 | 12 | 6 | 16 |

*Table 3: Requirements and high-level design outcomes.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |

*Table 4: Design outcomes.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 4 | 1 |
| Project Plan | 5 | 1 |
| Usability analysis | | |
| Requirements specification | 14 | 1 |
| Design plan | | |
| User interface document | | |
| Test plan | | |
| Test report | | |
| Usability test report | | |
| Final report | | |
| Project's story | 4 | 1 |
| Weekly reports | 10 | |

*Table 5: Documents.*

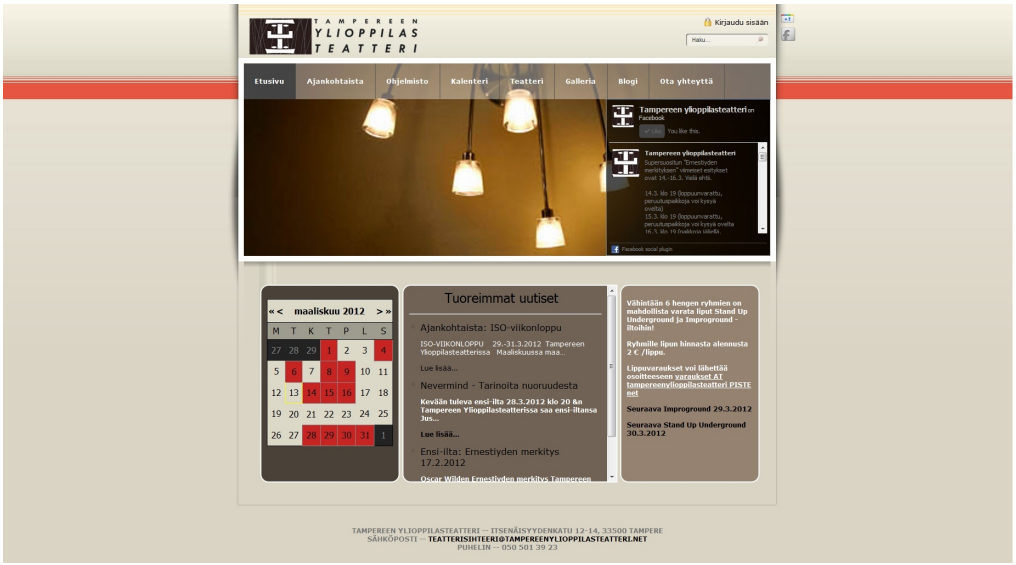
| Language | JAVA |
|---|---|
| LOC | 11341 |
| SLOC | |
| Reused code | 0 |
| Reused and modified | 0 |
| Classes | 166 |
| Functions | 1186 |
| Code revisions | |

*Table 6: Codelines.*

# Ylioppilasteatteri-projekti

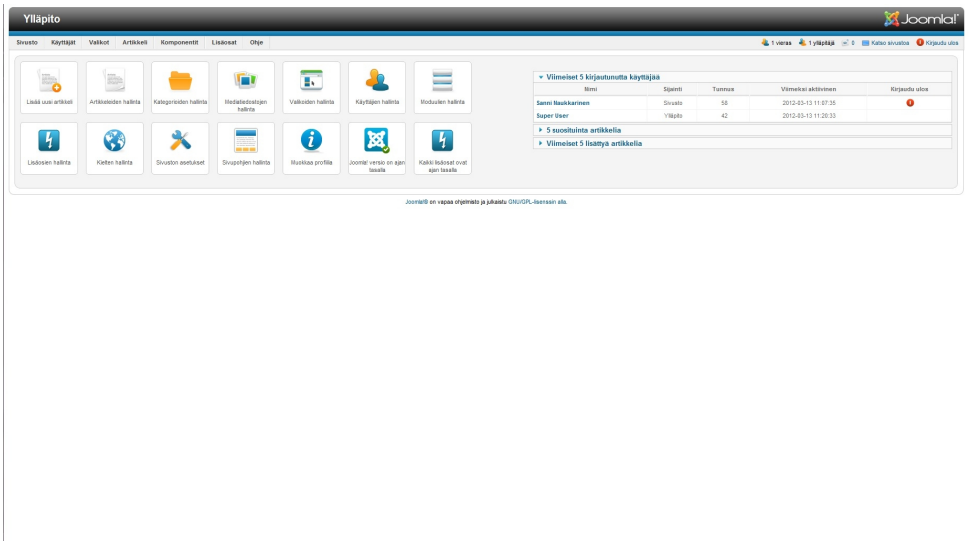
## Yleistä

Projektin lopullisena tuotoksena syntyi uusittu www-sivusto Tampereen ylioppilasteatterille. Tampereen ylioppilasteatteri on Tampereella toimiva harrastajateatteri, jolla perinteisen teatteritoiminnan lisäksi on improvisaatio- ja stand up -iltoja. Projektissa uudistettiin sivuston tekninen toteutus, toiminnot sekä visuaalinen ilme kaikkinensa. Sivusto on otettu jo käyttöön ylioppilasteatterin toimesta ja sen mahdollinen jatkokehitys ja ylläpito ovat ylioppilasteatterin hoidossa. Sivusto on katsottavissa osoitteessa www.tampereenylioppilasteatteri.net. Kuvassa 1 on uuden sivuston etusivu.




**Kuva 1. Etusivu.**

Kaikki sisällön hallinta blogia ja vieraskirjaa lukuun ottamatta hoituu admin-käyttöliittymän kautta.(Kuva 2).




**Kuva 2. Admin-käyttöliittymä.**

## Organisaatio ja projektinhallinta

Projektin asiakas oli Tampereen ylioppilasteatteri.

**Asiakkaan edustaja:**

Erkki Mervaala, joka oli myös osana projektiryhmää.

**Projektipäälliköt:**

Kari Jussila

Jari Laitinen

**Projektiryhmänjäsenet:**

Erkki Mervaala

Perttu Hallikainen

Massimo Prencipe

Miia Ketolainen

Teemu Pirinen

## Menetelmät ja työkalut

Projektin kotisivuina toimivat projektisivut Redmine-projektinhallintasovelluksessa, jossa pidettiin yllä lähes kaikkea projektiin liittyvää informaatiota ja tehtävienhallintaa. Tapaamisten pöytäkirjat, linkit työkaluihin, vinkit, dokumentit ja työtuntien seuranta olivat kaikki samassa portaalissa. Projektisivusto toimi pääosin moitteettomasti, työkaluna siinä ei ollut moittimista. Kommuniointiin ryhmä käytti facebookiin luotuo ryhmää, joka osoittautui varsin toimivaksi ratkaisuksi. Facebook ryhmän vahvuuksia olivat itsestään tapahtuva vanhojen keskustelujen dokumentointi ja käyttäjien helppo tavoitettavuus, kaikki ryhmäläiset kun käyttivät facebookia myös muihin tarkoituksiin.

Projekti toteutettiin Joomla sisällönhallintajärjestelmää käyttäen ja sen eri osista kooten. Lisäosia joutui toisinaan muokkaamaan paljonkin, mutta niiden perustoiminta oli aina valmiina ja saimme keskittyä enemmän niiden hiomiseen projektiin sopiviksi.

## Projektin vaiheet ja kehitysmalli

Projektissa käytettiin SCRUM-ohjelmistokehitysmallia, jossa pyrittiin kahden viikon sprintteihin. Muutamien sprinttien kohdalla tosin niiden pituudessa hieman joustettiin, koska tehtävät olivat hieman laajempia ja uusia tehtäviä ei varsinaisesti ollut tarjolla.

Projektin vaiheet sprinteittäin eriteltynä:

| Vaihe | Aloitus- ja lopetus pvm | Pituus | Kuvaus | Arvioidut työtunnit | Toteutuneet työtunnit |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Esitehtävät | 12.09.2011 - 19.09.2011 | Viikko | Työkalujen asennus, esitutkimuksen tekeminen | 83h | 85h |
| Sprint 1 | 20.09.2011 - 03.10.2011 | 2 viikkoa | Suunnittelu | 118h | 107h |
| Sprint 2 | 04.10.2011 - 17.10.2011 | 2 viikkoa | Suunnittelu ja toteutus | 118h | 75h |
| Sprint 3 | 18.10.2011 - 31.10.2011 | 2 viikkoa | Toteutus | 118h | 48h |
| Sprint 4 | 01.11.2011 - 14.11.2011 | 2 viikkoa | Toteutus | 118h | 73h |
| Sprint 5 | 15.11.2011 - 28.11.2011 | 2 viikkoa | Toteutus | 118h | 76h |
| Sprint 6 | 29.11.2011 - 16.12.2011 | 18 päivää | Toteutus ja testaus | 151h | 123h |
| | 17.12.2011 - 08.01.2012 | 23 päivää | Joululoma | 0h | 73h |
| Sprint 7 | 09.01.2012 - 23.01.2012 | 2 viikkoa | Toteutus ja testaus | 118h | 114h |
| Sprint 8 | 24.01.2012 - 12.02.2012 | 20 päivää | Toteutus ja testaus | 168h | 177h |
| Sprint 9 | 13.02.2012 - 09.03.2012 | 26 päivää | Testaus ja korjaus | 118h | 121h |

**Taulukko 1: Projektin sprintit**

Seuraavassa on listattuna projektin tärkeimmät virstanpylväät. Näissä vaiheissa projektista raportoitiin tarkemmin ulkopuolelle, sekä arvioitiin projektin tilannetta.

| **Tapahtumat** | **Pvm.** |
|---|---|
| Esitutkimus | 23.9.2011 |
| Projektisuunnitelma | 7.10.2011 |
| Katselmointi 1 | 14.11.2011 |
| Katselmointi 2 | 16.12.2011 |
| Katselmointi 3 | 30.1.2012 |
| Loppuesitys | 7.3.2012 |

**Taulukko 1: Projektin virstanpylväät**

## Johtopäätökset

Projekti onnistui kokonaisuutena hyvin. Kokemukset ovat ainakin vielä tätä kirjoitettaessa varsin positiivisia. Kaikki asiakkaan toivoma toiminnallisuus saatiin lopulta toteutettua ja ulkoasuakin ehdittiin hiomaan toiminnallisuuden ohella vastaamaan toiveita. Ainakin projektin toteuttavat oppivat uusia asioita ja saivat arvokkaan kokemuksen onnistuneen asiakasprojektin läpiviennistä.

Ensi kerralla paremmin voisi hoitaa projektin alussa työkalujen miettimisen ja versionhallinnan toteuttamisen. Paljon muokkaamista tuli tehtyä lopulta suoraan ftp yhteyden kautta palvelimella oleviin tiedostoihin ilman jatkuvaa versionhallintaa. Kyseinen käytäntö aiheuttaa aina pienen riskin vaikeasti korjattavien ja löydettävien ongelmien syntymiselle.

Ryhmä toimi mukavasti yhteen, eikä kukaan jäänyt porukasta ulkopuolelle. Palaverit pidimme yhtä viikko lukuun ottamatta joka viikko, joka kertoo hyvästä sitoutumisesta yleisellä tasolla. Lopulta myös projektin aluksi arvioitu kokonaistuntimäärä tuli lähes täytettyä, vaikka projektin aikana arvioimme jäävämme siitä. Jakautuminen ei kuitenkaan ollut suunnitellun tasainen, vaan tuntien haarukka ryhmän sisällä heitti yli sadalla tunnilla jäsenten kesken. Halukkaat saivat tehdä enemmän ja kiireisemmät hieman vähemmän.

Tarkka odotettujen riskien luettelu löytyy projektin projektisuunnitelmasta ja loppuraportista. Seuraavassa on lyhyesti esiteltynä projektin kohtaamat riskit.

**Käytettävyys**: Varsinaiset käytettävyystestaukset tulevilla sivuston käyttäjillä jäi projektin aikana suorittamatta. Käytettävyystestit eivät muutenkaan olleet ykkösprioriteettimme vaan olisimme suorittaneet testauksia siinä tapauksessa, mikäli ylimääräistä aikaa olisi jäänyt projektin loppuajasta runsaasti. Sivustomme sai kuitenkin paljon positiivista palautetta Tampereen Ylioppilasteatterin henkilöstöltä.

**Tuntematon tekniikka:** Joomla oli kaikille uusi julkaisujärjestelmä ja toimintaympäristö, joten itse järjestelmän käytön opettelu vei kaikilta projektiryhmän jäseniltä huomattavan paljon aikaa. Onneksi tähän oli varauduttu ajallisesti.

**Henkilökohtainen aika projektiin**: Monet ryhmän jäsenet kävivät päivätöissä ja suorittivat tietysti muita opiskeluja projektin kanssa samanaikaisesti, joten projektiin jokaviikkoinen täysi panostus oli mahdotonta. Tähänkin riskiin oltiin onneksi varauduttu ajallisesti ja projektipäälliköiden suomalla joustavuudella.

# Tilastot

## Projektin yleiset tiedot

| Ryhmän koko | Kehitysmalli | Aloituspäivä | Lopetuspäivä | Päiviä yht. | Tunteja yht. |
|---|---|---|---|---|---|
| 2+5 | SCRUM | 12.9.10 | 9.3.11 | 180 | 1188 |

**Taulukko 3: Projektin tiedot**

## Projektiryhmän työtunnit

| Toiminta | Projektin suunnittelu ja johtaminen | Vaatimusten määrittely | Suunnittelu | Ohjelmointi | Integrointi ja testaus | Katselmoinnit | Korjaus | Opiskelu | Muut | Yht. |
|---|---|---|---|---|---|---|---|---|---|---|
| Tunnit | 142 | 27 | 323 | 321 | 51 | 16 | 25 | 229 | 54 | 1188 |
| % | 12 | 2 | 27 | 27 | 4 | 1 | 2 | 19 | 5 | 100% |

**Taulukko 2: Projektiryhmän kokonaistunnit**

## Dokumentaatiot

| **Dokumentti** | **Sivut** | **Versiot** |
|---|---|---|
| Esitutkimus | 8 | 1 |
| Projektisuunnitelma | 18 | 5 |
| Suunnitteludokumentti | 8 | 5 |
| Vaatimustenhallintadokumentti | 28 | 15 |
| Loppuraportti | 14 | 5 |
| Projektikertomus | 6 | 1 |
| Viikkoraportit | 21+2 | - |

**Taulukko 3: Dokumentit**

# Asiaa! case management system

## Overview

Case management systems are used in the public administration for keeping track of the management processes. The case register has a central role in the case management systems, for it contains all the cases handled by the authorities, the documents involved in the process, as well as actions performed by different users in the case. It works as a tool for following all the actions taking place in the system and also shows what cases are open in the organization, and what cases are finished.

An example of a process controlled by a case management system would be a permission applying process for a hot dog stand. The process starts when somebody makes an initiative. The authorities receive the initiative and forward it to the person that will handle the case. The handler then asks for an opinion from the relevant opinion givers. The handler makes a proposition for decision and documents it. Finally the decision is made and the initiative maker is notified. All the phases, actions made in the process and documents created are saved in the case register.

The Asiaa! system simulates the essential functionality of a complete case management system. Its purpose is to act as an efficient teaching tool by which students get a concrete feel of how a case management system works for all the involved parties, and as a whole.

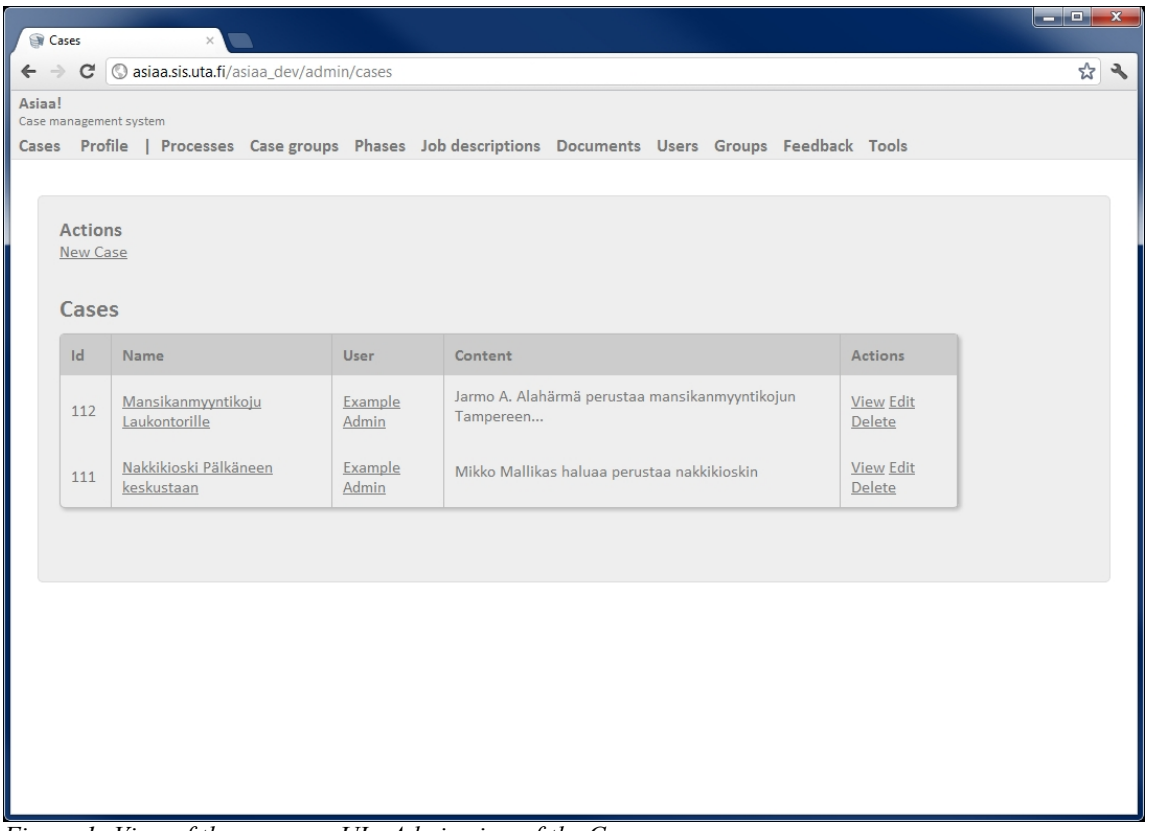


*Figure 1: View of the program UI - Admin view of the Cases*

## Organization and management

The project started with seven team members, one member dropped during the first weeks of the course because of other duties. The project staff includes the following members: Managing the project were Juhani Linna and Pompo Stenberg, and the programming and UI design was done by rest of the group: Aku Häsänen (programming), Lauri Maasola (programming/testing/documenting), Juho Sirén (programming/testing/documenting) and Antti Varjoinen (programming/UI design).

The project management model used in the project worked well in practice. Pompo handled mostly the managing of the technical aspects of the project and Juhani concentrated his efforts on the administrative tasks in the project.

## Methods and tools

The project used Subversion for handling the code repository. The members had different operating systems but everyone who was coding used NetBeans for editing the code. NetBeans proved to be an efficient development tool and its Subversion integration proved easy to use.

CakePHP (using PHP 5)was the framework of choice, as one project manager and worker had previous good experiences with it. The setup of CakePHP and all the needed services (enclosed in the Apache distributions LAMPP or XAMPP, depending on the operating system) caused for the inexperienced group members fair amount of work, but once the Apache distribution was up and running, everyone was able to do development work with CakePHP without larger technical problems.

Project tracking activities were done with Atlassian's JIRA software. All tasks were entered to JIRA by the managers and group members changed the status of the tasks themselves as soon as they were completed and ready for testing.

For communicating between the team members, web-based Flowdock software was used. It offered the functionality of the full version free of charge for students. The program was an effective tool for communication between the team members. Its two noteworthy functionalities are the automatic e-mail notifying for when someone's name is tagged in a discussion, and a file sharing functionality.

The team used a wiki for keeping track of the hours and sharing the important documents.

## Project phases and development model

The development model was an applied Scrum with two-week long sprints. Because of team members' time and resource restrictions, instead of daily meetings the team Scrum-meeting was held every week on Thursday. In addition, in most weeks an "evening school" was held at Tuesday afternoons where more technical issues were discussed and solved in a group. Table 2 shows the project sprints.

| Sprint | Date |
|---|---|
| Sprint -2 | 16.-29.9. |
| Sprint -1 | 30.9.-13.10. |
| Sprint 1 | 14.10.-27.10. |
| Sprint 2 | 28.10.-10.11. |
| Sprint 3 | 11.11.-24.11. |
| Sprint 4 | 25.11.-8.12. |
| Sprint 5 | 9.12.-13.1. |
| Sprint 6 | 14.1-25.1. |
| Sprint 7 | 26.1.-9.2. |
| Sprint 8 | 10.2.-8.3. |

*Figure 2: Sprints of the project*

The following table lists the project milestones.

| Milestone | Date |
|---|---|
| Preliminary Analysis Review | 22.9. |
| Project Plan Review | 10.10. |
| 1st Sprint Review | 27.10. |
| 2nd Sprint Review | 10.11. |
| 3rd Sprint Review | 24.11. |
| Midterm Presentation | 30.11. |
| 4th Sprint Review / December review | 9.12. |
| 5th Sprint Review | 13.1. |
| Final Review | 10.2. |
| Final Presentation | 14.3. |

*Figure 3: Milestones of the project*

## Experiences

The project turned out to be a success. It helped, that the project managers had experience from real-life software projects, and also some of the group members had earlier technical experience with the framework of choice, CakePHP. One group member quit the project in an early phase, because of lack of time. The commitment of the group members was generally good but not very even: The developers with

previous experience with PHP were more willing to work long hours to satisfy the requirements set to the project. The participation in weekly meetings was in a good level and the group communication worked quite well through FlowDock, which turned out to a valuable tool for the group.

The biggest challenges were related to the requirements of the project, as these were not always clear even to the group managers. Because of the non-existence of similar software, only when the project was at a mature enough state, it could be demoed to the client. And only then was the client able to see some deficiencies in the software. But generally those deficiencies were not too large, so after some fixes all the major project goals were achieved and almost in the planned, optimistic timeframe!

## Statistics

| Team size | Dev. model | Start date | End date | Days | Hours |
|---|---|---|---|---|---|
| 2+4 | Scrum | Sep 9 | Mar 9 | 182 | 972 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 260 | 14 | 16 | 268 | 80 | 19 | 26 | 209 | 82 | 972 |
| % | 27 | 1.4 | 1.6 | 28 | 8.2 | 2 | 2.6 | 22 | 8.4 | 100 |

*Table 2: Group effort by activity.*

| Number of requirements | Database diagrams | Database tables |
|---|---|---|
| 37 | 2 | 15 |

*Table 3: Requirements and high-level design outcomes.*

| Language | PHP | Shell | CSS | Javascript |
|---|---|---|---|---|
| Files | 58 | 2 | 1 | 3 |
| LOC | 3773 | 507 | 388 | 34 |
| SLOC | 4986 | 545 | 489 | 42 |
| Code revisions | 409 | | | |

*Table 4: Codelines, framework excluded.*

| Document | Pages |
|---|---|
| Assignment Document | 2 |
| Preliminary Plan | 7 |
| Project Plan | 13 |
| Test plan | 4 |
| Final report | 12 |
| Project story | 6 |
| Persona Documents | 3 |
| Weekly reports | 133 (.ppt slides) |

*Table 5: Documents.*

# Editaattori

## Overview

Editaattori is an educational tool for teaching media text producing and layout. It is built for high school and university use. Students can do exercises and exams about inserting titles, headings, pictures etc to pre-made media texts. Teachers can create these exercises and grade them.

Our customer was Maarit Jaakkola who is a lecturer of journalistics at University of Tampere. She had an assigment from the Finnish National Board of Education to search for new eMethods for studying.

From the very beginning of the project it was agreed that we would not aim to create a production ready product. The goal was to create a prototype that the next project team could improve on later.

*Illustration 2: Editaattori Frontpage.*

*Illustration 3: Student Exercise View.*

## Organization and management

1. Mika Kähkönen, project manager
2. Jukka Similä, project manager
3. Jussi Kivinen, project manager
4. Petri Arpiainen, developer
   - During the project Petri was mainly involved in the feature programming process and in the bugfixing. Of the features in the product, he was primarily working on the module that handles single exercises
5. Anna Kamunen, developer
   - In the project Anna was involved with coding. Her tasks had to do with the functionality of maintaining courses. She also participated in requirements definition and documentation.
6. Anu Kauppi, chief testing officer, documentation
   - Anu was responsible for documentation, test planning and testing. She wrote the test plan and also parts of other documents. She also participated in usability testing in Sammon keskuslukio. She tested Editaattori several times during the project.
7. Miikka Kähkönen, developer
   - Miikka was involved mostly in programming. He focused on the features that the teachers are using.
8. Teemu Mikkola, database and documentation manager
   - Teemu’s main role in the project was to concentrate on the program's usability and testing. He was mainly responsible of the program's requirement specification document and also wrote to many other documents.


*Illustration 4: Project team.*

## Experiences

I was a project manager. Half of my work was planning and management. I also got familiar with a new framework to me, Joomla, and participated in the coding, testing and repairing. It was one-third of my work. The project was satisfying: I had the opportunity to try to delegate tasks, communicate with client, do requirement specification, deal with project members etc. I am pleased with the results, as we had challenges with schedules both with the client and group members. That resulted in almost every group member getting minimum amount of hours

– **Mika Kähkönen**

My role as one of the three project managers was 100% managing and supervising. I was involved with outer and inner communication, meeting organising, leading the meetings, creating documentation, timetables etc. It was interesting and educational to participate in software project with so many new persons.

– **Jussi Kivinen**

My role was to be a Joomla specialist, since I was the only person who knew the framework in advance. However, I tried to avoid getting too involved with the coding and tried to focus on project management, attempting to keep the overall amount of work reasonable and delegating the work among team members in cooperation with other project managers. For me the most important lesson was to let others do the programming and concentrate on supervising the work.

– **Jukka Similä**

## Methods and tools

### Tools

Editaattori is web-based program that can be used with normal web browsers. The application is built on top of the Joomla content management system.

Most of the team used Zend Server CE and Eclipse PDT with Subversion-plugin running on Windows. One member used XAMPP. One member had Ubuntu with RapidSVN.

Communication occurred mostly by Facebook and partly by UTA email. At the beginning of the project we attempted to use IRC for realtime communication, but since the participation was low, we decided to use Facebook instead, which was a successful decision. Documents were stored in Google Docs and Redmine, which was also used for task management and for hosting our wiki page.

Our tools suited projects purposes and needs just fine.

### Methods

For the software project we used Scrum methodology with some modifications. We had no resources to have daily Scrum meetings, so we had weekly meetings on Mondays. At these meetings we discussed the duties of the previous week and agreed about next tasks to be done.

*Illustration 5: Team in the weekly meeting.*

## Project phases and development model

The project had three separate phases; Planning, building and finish/documentation phase. All phases consisted of two to three sprints. After every sprint there was a sprint review and after every phase there was a phase review with the team, the customer and the course teacher.

**Project Timetables**

| | |
|---|---|
| Planning phase | 1.9.–10.10. |
| First project meeting | 13.9. |
| Project Plan | 10.10. |
| Phase 1 | 10.10–20.10. |
| Sprint 1 | 10.10.–23.10. |
| Sprint 2 | 24.10.–6.11. |
| Sprintti 3 | 7.11.–20.11. |
| Personal report I | 1.–7.11. |
| First review | 21.11. |
| Phase 2 | 21.12.–8.1. |
| Midterm presentations | 30.11. |
| Sprint 4 | 21.11.–16.12. |
| second review | 21.12. |
| Sprint 5 | 19.12. – 4.1. |
| Phase 3 | 9.1.–12.2. |
| Personal report II | 1.1. –16.1. |
| Sprint 6 | 9.1.– 26.1. |
| Sprint 7 | 30.1.– 12.2. |
| Third review | 30.1. |
| Documentation and bug fixing | 12.2.–9.3. |
| Final report | 5.3. |
| Final meeting | 6.3. |
| Project story | 9.3. |
| Project CD | 9.3. |
| Final presentations | 14.3. |
| Personal report III | 03.2012 |

## Statistics

| Team size | Dev. model | Start date | End date | Days | Hours |
|---|---|---|---|---|---|
| 3+5 | Scrum | 13.9.2012 | 14.3.2012 | 184 | 1300 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 513.75 | 54.25 | 54.50 | 245.25 | 78 | 57.25 | 34.75 | 114 | 66 | 1217.75 |
| % | 42.2 % | 4.7 % | 4.5 % | 20.1 % | 6.4 % | 4.7 % | 2.9 % | 9.4 % | 5.4 % | 100% |
| Total | | | | | | | | | | **1217** |

*Table 2: Group effort by activity.*

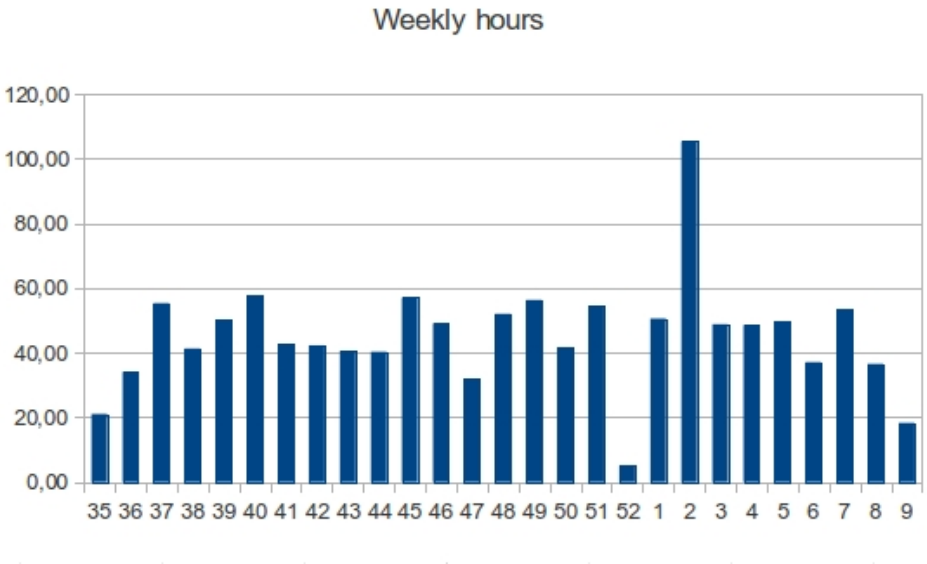



*Illustration 6: Weekly hours*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 64 | 23 | 2 | 4 | 1 | 44 |

*Table 3: Requirements and high-level design outcomes.*

| Week | Hours |
|---|---|
| 2011-35 | 21,00 |
| 2011-36 | 34,00 |
| 2011-37 | 55,25 |
| 2011-38 | 41,25 |
| 2011-39 | 50,25 |
| 2011-40 | 57,75 |
| 2011-41 | 42,75 |
| 2011-42 | 42,25 |
| 2011-43 | 40,50 |
| 2011-44 | 40,25 |
| 2011-45 | 57,00 |
| 2011-46 | 49,00 |
| 2011-47 | 32,00 |
| 2011-48 | 52,00 |
| 2011-49 | 56,25 |
| 2011-50 | 41,50 |
| 2011-51 | 54,50 |
| 2011-52 | 5,00 |
| 2012-1 | 50,50 |
| 2012-2 | 105,25 |
| 2012-3 | 48,75 |
| 2012-4 | 48,50 |
| 2012-5 | 49,50 |
| 2012-6 | 37,00 |
| 2012-7 | 53,50 |
| 2012-8 | 36,50 |
| 2012-9 | 18,25 |
| Total | 1220,25 |

*Table 4: Weekly hours*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 5 | 1 |
| Project Plan | 22 | 19 |
| Requirements specification | 23 | 5 |
| Test plan | 26 | 9 |
| Test report | 18 | 1 |
| Usability test report | 17 | 1 |
| Final report | 23 | 7 |
| Project's story | | |
| Weekly reports | 25 reports | |
| User manual | 8 | 1 |

*Table 5: Documents.*

| Language | PHP, JavaScript, SQL |
|---|---|
| LOC | PHP 4493, JS 170, CSS 300 |
| SLOC | |
| Reused code | Joomla: PHP >220000, JS >70000<br>Artisteer: PHP 2189 |
| Reused and modified | Artisteer: CSS 3800 |
| Classes | 41 (+ reused) |
| Functions | 159 (+ reused) |
| Code revisions | 222 |

*Table 6: Codelines.*

# Shakkilinna

## Overview

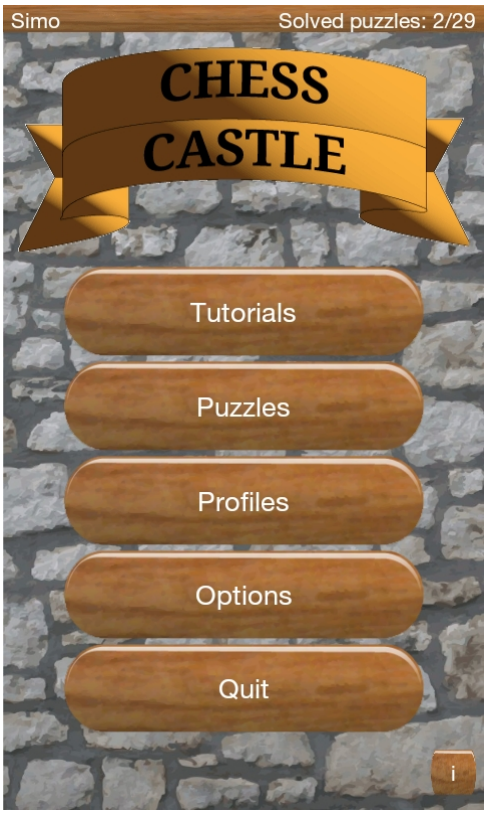
Shakkilinna was a project that aimed to create a chess learning application for Android mobile devices. The project was established as a combination of courses on Software Project Management and Project Work, given at the Department of Computer Sciences at the University of Tampere.

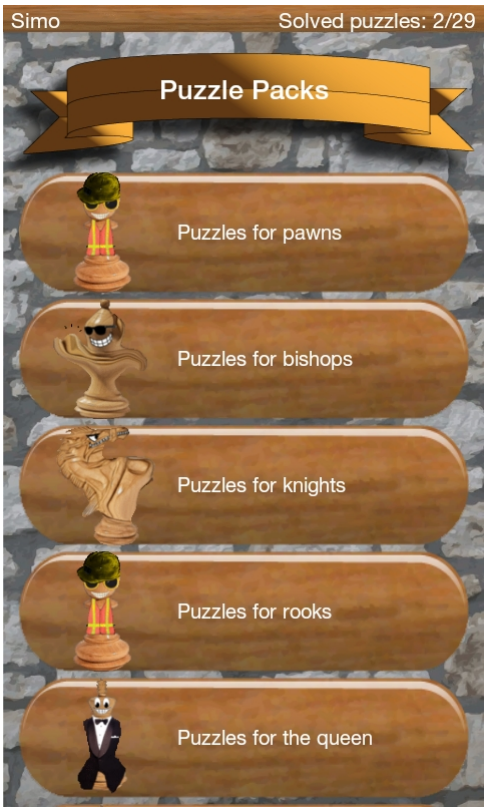
The project was inititated by the client, Shakkilinna, in September 2011. The project was finished in March 2012.

## The look and feel of the Chess Castle application

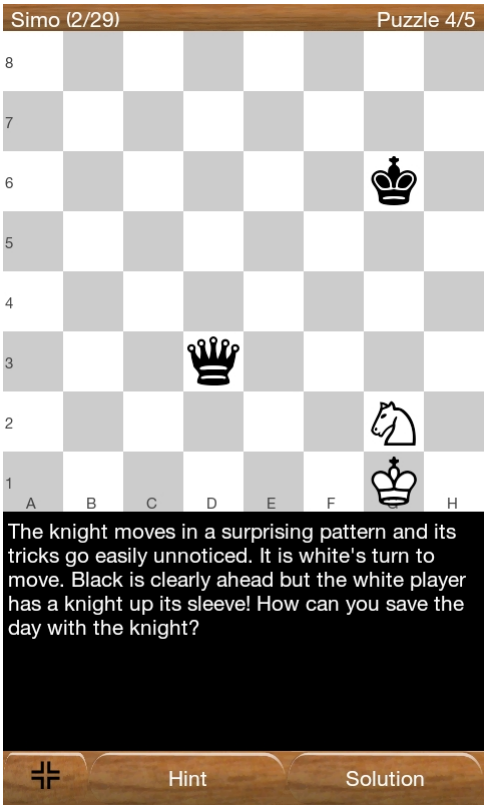
Chess Castle's name is derived from the name of the project's client, Shakkilinna. The general appearance of the application's user interface follows the theme set by the name.

The client provided additional graphics to display on the puzzle pack list as seen in the screen capture below on the right.



The chess board uses an open chess font to display chess pieces. The chess board itself is programmatically drawn as are the highlighted squares.





## Organization and management

The project group consisted of two managers and five other project members with varying tasks.

### Project Managers

Ville Mäkelä and Simo Pönni

### Project Group

Miika Haverinen

- Programming and design

Jani Mikkola

- Programming and UI design

Jyri Mäkinen

- Usability design and programming

Ville Riikonen

- Programming and design

Juho Ylipoti

- Programming and design

The project group had weekly meetings, during which the tasks for the past week were reviewed and new tasks were assigned. Group members also had a four-hour workshop every week, in which they could work and learn effectively as a group. A

web-based instant messaging software, Flowdock, and email were used outside weekly meetings and workshops to discuss various project-related tasks.

## Methods and tools

The development tools used in the project were decided beforehand and set up in a meeting with the group. There were no changes in the tools used during the project, although Redmine's use declined. In the end, Redmine was used mainly for hour reporting.

**Development tools and programming languages**

Eclipse SDK, version 3.71
Java JDK, version 1.7.0
Android SDK Tools, revision 13
Android SDK Platform Tools, revision 7
SDK Platform Android 2.1-update1, API 7, revision 3

**Database**
SQLite

**Version control**
Subversion, version 1.6.17 (r1128011)

**Task and time management**
Redmine https://redmine.sis.uta.fi/
Flowdock, browser-based instant-messaging with conversation logging and file uploads https://shakkilinna.flowdock.com/

**Documentation**
Project course wiki: https://projectwiki.cs.uta.fi/wiki/Shakkilinna
Google Docs was used for writing all reports and planning documents.

**Testing tools**
No tools for specifically testing code were used.

## Project phases and development model

Most of the projects milestones were met on time or with acceptable deviation from expected, as can be seen in the table below.

The technical implementation could not be stopped as planned in mid-February. Nevertheless, the group felt that the actual implementation was done – what was left was only repair and aesthetic corrections.

| Event | Est. date | Realized date |
|---|---|---|
| Project plan review | 17.10.2011 | 17.10.2011 |
| First project review with client and project supervisor | 31.10.2011 | 31.10.2011 |
| Project presentations | 30.11.2011 | 30.11.2011 |
| Second project review with client and project supervisor | 12.12.2011 | 12.12.2011 |
| Technical implementation done | 12.2.2012 | 9.3.2012 |
| Final project presentations | February 2012 | 14.3.2012 |
| Project finished | 9.3.2012 | 11.3.2012 |
| Project's end review | 12.3.2012 | 12.3.2012 |

The project was planned originally to consist of six sprints. At some point, the lenghts of sprints were changed, because we realized that one sprint was unproportionately long compared to others and a review was needed in between. Otherwise, the schedule set in the beginning held well.

| | |
|---|---|
| Sprint 1: 3.10 - 31.10. | • Getting to know the project, group members and the client<br>• Requirements specification<br>• Familiarizing the group with development tools and setting up the development environment<br>• UI design<br>• UI sketches<br>• Technical design<br>1. Designing a chess puzzle presentation format<br>2. Starting to develop a prototype |
| Sprint 2: 31.10. - 21.11. | 1. UI design<br>2. Heuristic evaluation of the first UI<br>3. Puzzles<br>a. Hard-coded puzzles for evaluation, proof of concept<br>b. Mechanics for moving chess-pieces, solving puzzles<br>c. Puzzle importing from XML |
| Sprint 3: 21.11. - | 1. Technical implementation continued<br>a. Puzzle categorization |

| | |
|---|---|
| 12.12. | b. User profiles<br>c. Puzzle importing from XML<br>d. Localization<br>e. Saving user's progress<br>f. Moving between puzzles<br>2. Usability testing / -evaluation (heuristic)<br>3. Program architecture planning<br>4. The following requirements must be done by the end of the sprint<br>a. Must be able to create puzzles<br>b. User must be able to solve puzzles<br>c. User must receive feedback from completing a puzzle<br>d. User must receive feedback from failing to complete a puzzle<br>e. Finishing tasks that were begun in sprint 2<br>i. Must be able to select puzzle packs and puzzles<br>ii. User must be able to see a puzzle description at the start of the puzzle<br>5. The first version of usability test plan |
| Period break: 20.12. - 8.1. | |
| Sprint 4: 9.1. - 30.1. | 1. Finalizing the features of the software<br>2. Technical implementation continued<br>a. Moving between puzzles<br>b. Saving the user's progress<br>c. Integration of graphics<br>d. Localization<br>3. Technical testing<br>4. Big fixing<br>5. Finalized usability test plan<br>6. Usability testing<br>7. The following requirements must be done by the end of the sprint<br>a. User must be able to change profiles while running the software<br>b. User must be able to select all previously completed puzzles and the first not-completed puzzle<br>c. User must be able to move on to the next puzzle<br>d. User must be able to view the puzzle description and the puzzle hint by pressing a button |

| | |
|---|---|
| | e. User must be able to maximize the description window<br>f. A profile's progress through puzzles must be saved into the database<br>g. The software needs to maintain its state unless it is intentionally closed<br>h. Finishing tasks that were begun in sprint 2<br>i. Must be able to select puzzle packs and puzzles<br>ii. User must be able to see a puzzle description at the start of the puzzle<br>8. Focus on quality requirements<br>a. Quality requirement: Quality of information<br>b. Quality requirement: Technical quality<br>c. Quality requirement: Usability<br>d. Quality requirement: User Interface |
| Sprint 5:<br>30.1. - 20.2. | 1. Usability fixes<br>2. Technical testing<br>3. Bug fixing<br>4. The following requirements must be done by the end of the sprint<br>a. User must be able to see his/her progress throughout the software<br>b. User must be able to change the settings while running the software<br>c. When starting the software for the first time, the language must be selected based on the language of the running device<br>5. Focus on quality requirements<br>a. Quality requirement: Quality of information<br>b. Quality requirement: Technical quality<br>c. Quality requirement: Usability<br>d. Quality requirement: User Interface |
| Sprint 6:<br>20.2. - 9.3. | 1. Bug fixing/Other fixes<br>2. Review of software quality<br>3. Documentation of the software and the project<br>4. Handing out the materials to the client<br>5. Briefing of the client<br>6. Finishing the project 9.3. - 14.3.<br>a. Project story ready 9.3.<br>b. Final report ready 9.3.<br>c. Final meeting 12.3. |

## Experiences

Project members feel that the project was a success – a thought shared by the client and the course supervisor. The project was felt to be interesting and challenging, and it also provided the team with experience from various fields. The end product also serves as a work example in the future.

The client can make use of the end product in chess teaching, use it to promote chess in general, and – with a little effort – commercialize the product for profit.

**Risks met during the project**

Motivational Problems

During December of 2011, project members had a lot of course works and exam to take care of, which resulted in a brief lack of motivation for this particular project. Project members had a two-week holiday, during which no progress for the project was made, which helped people to re-acquire their motivation for the starting year.

Project members getting sick

Project members getting sick was not a serious issue, although some tasks were briefly delayed because of this.

Timetable problems

Both of the project managers were also working during the project, which prevented them from taking part in the project as much as they would have wanted. Issues we solved, although it required a great deal of effort from time to time.

## Statistics

In this section we report our project's statistics. The tables and charts below present key figures about time spent, categorized by type of effort or by time period.

The section also covers information about the products of the project: Types of documentation and the number of required iterations to achieve them and a breakdown of produced code of the resulting application.

| Team size | Dev. model | Start date | End date | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | Scrum But | 1.9.2011 | 16.3.2012 | 198 | 1312 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 373,25 | 19 | 46 | 513,75 | 64,5 | 69 | 11 | 102,75 | 112,75 | **1312** |
| % | 28 | 1 | 3 | 39 | 5 | 7 | 1 | 8 | 8 | 100% |

*Table 2: Group effort by activity.*

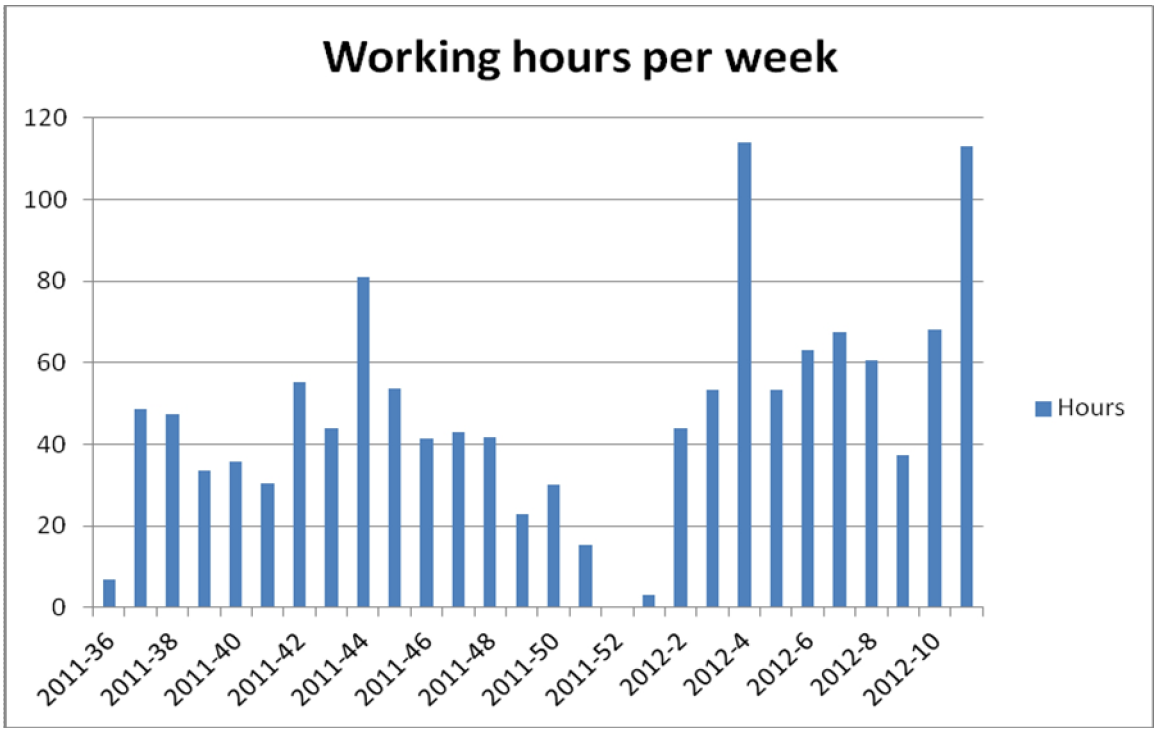



*Chart 1: Working hours per week*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 23 | 20 | 16 | 11 | 0 | 3 |

*Table 3: Requirements and high-level design outcomes.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 |

*Table 4: Design outcomes.*

| | |
|---|---|
| Language | JAVA |
| LOC | 5700 |
| SLOC | 5200 |
| Generated code (Eclipse / gen) | 225 |
| Created files | 45 |
| Chess puzzle XML-files | 35 |
| *Code revisions* | 267 |

*Table 5: Codelines.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 6 | 2 |
| Project Plan | 23 | 10 |
| Use Case design document | 10 | 5 |
| User interface design documentation | 10 | 5 |
| Requirements specification / Backlog | 4 | 7 |
| Use Cases | 2 | 3 |
| Development environment setting and code conventions | 3 | 2 |
| Usability test plan | 5 | 3 |
| Usability test – Script | 5 | 2 |
| Usability test – Background information form | 1 | 1 |
| Usability test report | 7 | 2 |
| Weekly reports | 1-2 pages, 26 pieces | 1 |
| Other meeting minutes | 1-2 pages, 6 pieces | 1 |
| Technical documentation | 20 | 5 |
| Puzzle editor usage guide | 4 | 3 |
| Final report | 27 | 3 |
| Project's story | 12 | 3 |

*Table 6: Documents.*

# Math.fi

## Overview

The object of the project was to develop the learning management system for mathematics.

The screenshot about the site:


## Organization and management

Project manager Tero Strakh, scrum master

project manager Pasi Kiema, project owner

project worker Hanne Korhonen

project worker Teemu Keskinen

project worker Arto Laurila

project worker Antti Leppänen

project worker Sami Koivunen

Weekly face to face meetings were the base of work. We had also weekly scrum meeting on IRC. IRC was used every day for keeping contact.

JIRA was used for engineering requirements (backlogs) and the bookkeeping about the working hours was also there.

### Methods and tools

Developing environment: LAMP (Linux) or XAMPP (Windows)

Database: MySQL

IDE: Netbeans

Documents and notes: wiki (wiki.math.fi)

Project management: JIRA + GreenHopper

Communication: IRC, email, phone, face to face meetings.

### Project phases and development model

We used applied scrum as a development model. Daily scrums were replaced with an IRC meeting once a week and a face to face meeting also once a week. Informal meetings were held in IRC almost every day.

Planning took place from 20th of September to 11th of October when the review of the project plan was held.

We had four sprints for implementing requirements. The reviews of the sprints were:

8th of November

7th of December

17th of January

16th of February

The final sprint was for testing, documenting and fixing bugs. It finished 15th of March when the final report was reviewed.

Notes about each weekly meeting are available in http://wiki.math.fi/index.php?title=Kokoukset

## Experiences

Absorbing technologies: basically everybody in the group could learn enough using the equipment and methodologies for working efficiency. Web-socket technique was unforeseen technology which we weren't able to adopt adequately enough during the project.

Late delivery from the graphic artists was unforeseen risk. But by increasing effort at the end of the project, we were able to implement the new layout of the site.

Access control was an expected risk which was almost met. In the beginning, it seemed to us that the access control was in order. Just at the end of the project we could find the bug concerning ACL. Fortunately the bug was fixed.

The update failed in February. Mostly, it was a result of lack for documentation. Even bigger problem was that there was just one database for developing and testing. Therefore we added one database for testing.

We did some useless work because the requirement specification had to be changed and some of the changes concerned also parts which were already done.

## STATISTICS

The statistics are made according to the state of 11th of March 2012. During the week 12th of March to 16th of March there's still some work left, approximately 80 hours. Those tasks include among other things, the delivery of the project to the client, fixing bugs and writing the documents.

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | scrum | 20.9.2011 | 11.3.2012 | 174 | 1200 |

*Table 1: General project information.*

| Week | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 15 | 39 | 49 | 44 | 46 | 24 | 12 | 36 | 48 | 26 | 69 | 52 | 32 | 30 |
| Week | 50 | 51 | 52 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | **Total** |
| Hours | 50 | 25 | 25 | 25 | 70 | 72 | 52 | 44 | 55 | 69 | 59 | 63 | 70 | **1200** |

*Table 2: Weekly working hours, rounded in integer.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 298,33 | 24 | 161,75 | 380,42 | 33 | 42,5 | 25,5 | 103,5 | 51 | **1120** |
| % | 27 | 2 | 14 | 34 | 3 | 4 | 2 | 9 | 5 | 100% |

*Table 2: Group effort by activity. The statistics are taken from JIRA project management tool. First weeks (mostly study and the preparing of the project by managers are not marked in JIRA. That's the reason for the contradiction in the total hours.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| some | - | - | - | - | some |

*Table 3: Design outcomes.*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 37 stories in JIRA; not all implemented | ? | 37 stories in JIRA; not all implemented | Made by graphic artists (external stakeholders) .. | (Made afterwards.) | 36 |

*Table 4: Requirements and high-level design outcomes.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 7 | 1 |
| Project Plan | 15 | 2 |
| Requirements specification | In JIRA as stories | |
| User interface document | From graphic artists | |
| Final report | 13 | 1 |
| Project's story | 4 | 1 |
| Weekly reports | 24 | |

*Table 5: Documents.*

| Language | PHP |
|---|---|
| LOC | 4121 in controllers |
| SLOC | ? |
| Reused code | ? |
| Reused and modified | ? |
| Classes | ? |
| Functions | ? |
| Code revisions | 443 |

*Table 6: Codelines. The project continued from previous term. There is the amount of code in controllers for giving some advice about the expanse of the project. The estimate of lines made during this project is about half of total amount. In models there are quite few new lines but in views there're a lot.*

# Retu-järjestelmä

## Yleiskuvaus

Projektissa toteutimme resurssivaraus- ja tuntikirjausjärjestelmän Päivölän kansanopistolle. Järjestelmän tarkoituksena on:

- Hallinnoida Päivölän sisäisiä resurssivarauksia, kuten majoitus- ja opetustiloja sekä laitteita.
- Hallita opettajien lukujärjestyksiä ja seurata tuntimääriä.
- Lisäksi järjestelmän tuli olla laajennettavissa kattamaan myös kurssivaraukset ja opiskelijarekisteri.

Vaikka järjestelmä tuotetaankin Päivölän kansanopistolle, tarkoituksena oli tuottaa yleiskäyttöinen ohjelmisto, jota voidaan hyödyntää muissakin ympäristöissä.


Kuva 1. Järjestelmän etusivu.

### Organisointi ja hallinta

Projektiryhmä koostui kahdesta projektipäälliköstä sekä viidestä projektityöntekijästä.

Päälliköinä toimivat:

- Jussi-Antti Salomaa
- Jari Rantanen

Projektityöntekijöinä olivat:

- Katri Kaul

- Marjo Nopola
- Tuomo Hyttinen
- Markus Salomaa
- Teemu Jyrkämä

Projektipäälliköiden työt jakaantuivat siten, että Jari hoiti dokumentointia ja projektin yleisten asioiden hoitamista. Jussi-Antti hoiti viikkopalavereiden ja katselmointien pitämisen, yleisten asioiden hoitamista, projektiryhmän opastamisen sekä osallistui koodin tuottamiseen.

### Menetelmät ja työkalut

Järjestelmä tulee käyttöön Päivölän omalle LAMP-palvelimelle. Palvelimelle voidaan tarvittaessa asentaa tarvittavia ohjelmistokirjastoja tai frameworkeja.

Ohjelmointityökaluna käytettiin Eclipseä, johon integroitiin svn-versionhallinta. Ohjelmointikielenä toimi PHP ja lisäksi käytettiin ajax:ia ja javascriptiä. Sivujen ulkoasua muokattiin CSS-tyylien mukaan.

Projektiin liittyviä dokumenttejä ja ohjeita keräsimme Redmineen, josta ne ovat kaikkien saatavilla koko ajan. Viikkoraportointien julkaisu suoritettiin sähköpostin välityksellä. Raportit välitettiin jokaisen viikon maanantaina koko projektiryhmälle, kurssin vetäjille Timo Poraselle ja Pekka Mäkiaholle sekä asiakkaan yhteyshenkilölle Jooel Korvelle. Projektin yleinen dokumentointi suoritettiin LibreOfficella. Myös tuntikirjanpidon pitäminen onnistui Redminessä, joka osoittautui varsin hyväksi vaihtoehdoksi.

Projektiryhmän välinen kommunikointi hoitui Flowdock-keskustelukanavan kautta. Yleisen keskustelun lisäksi tämän kanavan kautta hoitui myös toisten ryhmäläisten auttaminen projektiin liittyvissä asioissa sekä esimerkiksi tapaamisten sopiminen.

### Projektin vaiheet ja kehitysmalli

Projektissa käytimme Scrum-kehitysmallia, jota kuitenkin muokkasimme hieman ryhmälle sopivaksi. Päivittäiset tapaamiset vaihdoimme viikkopalavereiksi, jotka projektin alussa pidimme lähes poikkeuksetta maanantaisin. Projektin edettyä vaihtelimme päivää sen mukaan, mikä sopisi kaikille parhaiten. Koko projektin ajan viikkopalaverit toimivat moitteettomasti ja koko ryhmä saatiin kasaan lähes joka kerta.

Työn jako suoritettiin projektin alussa melko järjestelmällisesti. Tietty henkilö vastasi tietystä osa-alueesta. Projektin edetessä työnjakoa hieman löysättiin ja yhdessä sovittiin kuka tekee mitäkin. Tämä osoittautui hyväksi valinnaksi, sillä ryhmän hyvä yhteishenki antoi tähän mahdollisuuden. Jokainen oli valmis auttamaan muita omien töidensä lisäksi.

| Päivämäärä | Vaihe |
|---|---|
| 14.9.2011 | Projektin esittely |

| | |
|---|---|
| 29.9.2011 | Esitutkimuksen vaiheistus |
| 6.10.2011 | Projektisuunnitelman katselmointi |
| 28.10.2011 | 1. katselmointi |
| 30.11.2011 | Projektin väliesitys |
| 17.11.2011 | 2. katselmointi |
| 19.12.2011 | 3. katselmointi |
| 17.2.2012 | Implementointi valmis |
| 14.3.2012 | Projektin loppuesitys |
| 16.3.2012 | Loppuraportti |

**Kokemuksia**

Software Project Management- sekä project work –kurssit koettiin erittäin haasteellisina, mutta tarpeellisina. Kurssit opettivat jokaiselle ryhmäläiselle paljon uusia asioita, joita voimme hyödyntää tulevissa opinnoissa sekä työelämässä. Projekti vaati paljon aikaa, mutta koimme ryhmämme siinä mielessä poikkeuksellisena, että kukaan ei jättänyt projektia kesken eikä luovuttanut missään vaiheessa. Projektin aikana lähes jokainen ryhmäläinen kävi muitakin kursseja saman aikaisesti, joten tämäkin aiheutti hieman vaikeuksia ajankäytön suhteen.

Kurssilla toteutunut projekti oli monelle ryhmäläiselle ensimmäinen hieman pidempi kestoinen projekti, joten sen aikana on saatu paljon merkittävää tietoa ohjelmistoprojektin läpiviemisestä. Mitä tavoitteiden saavuttamisen eteen on tehtävä, paljonko se vie aikaa, töiden jakamista sekä paljon yhteystyötä. Jokainen on päässyt toteuttamaan itseää ryhmässä ja saanut tehdä töitä sellaisten asioiden parissa, joita ei aiemmin ole tehnyt.

Monen mielestä olisi ollut hyvä, jos olisi käynyt ennen tätä projektia www-ohjelmointi –kurssin. Tämä olisi helpottanut projektin aloittamista, eikä olisi tarvinnut käyttää niin paljon aikaa menetelmien ja työvälineiden opetteluun.

Projektisuunnitelmassa esitellyt tavoitteet täyttyivät kokonaan, joten tulokseen voidaan olla tyytyväisiä. Lisäksi mieltä kohottaa se, että myös asiakas oli tyytyväinen lopputulokseen.

**Tilastotietoa**

| Ryhmän koko | Kehitysmalli | Aloituspäivä | Lopetuspäivä | Päiviä yhteensä | Tunteja yhteensä |
|---|---|---|---|---|---|
| 2+5 | Scrum | 5.9.2011 | 16.3.2012 | 194 | 1350,3 |

*Taulukko 1: Yleistä tietoa projektista.*

| Aktivit eetit | Plannin g and manag ement | Req. specifi cation. | De-sign | Code | Integration and testing | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|
| Tunnit | 564,35 | 64,5 | 77,5 | 368,95 | 32,25 | 39 | 161,25 | 42,5 | 1350,3 |
| % | 41,79 | 4,78 | 5,74 | 27,32 | 2,39 | 2,89 | 11,94 | 3,15 | 100 |

*Taulukko 2: Projektiryhmän kokonaistunnit.*

| Dokumentti | Sivumäärä | Versio |
|---|---|---|
| Esitutkimus | 7 | 1 |
| Projektisuunnitelma | 24 | 1.1 |
| Loppuraportti | 25 | 1 |
| Viikkoraportit | 25 | |
| Projektikertomus | 6 | 1 |

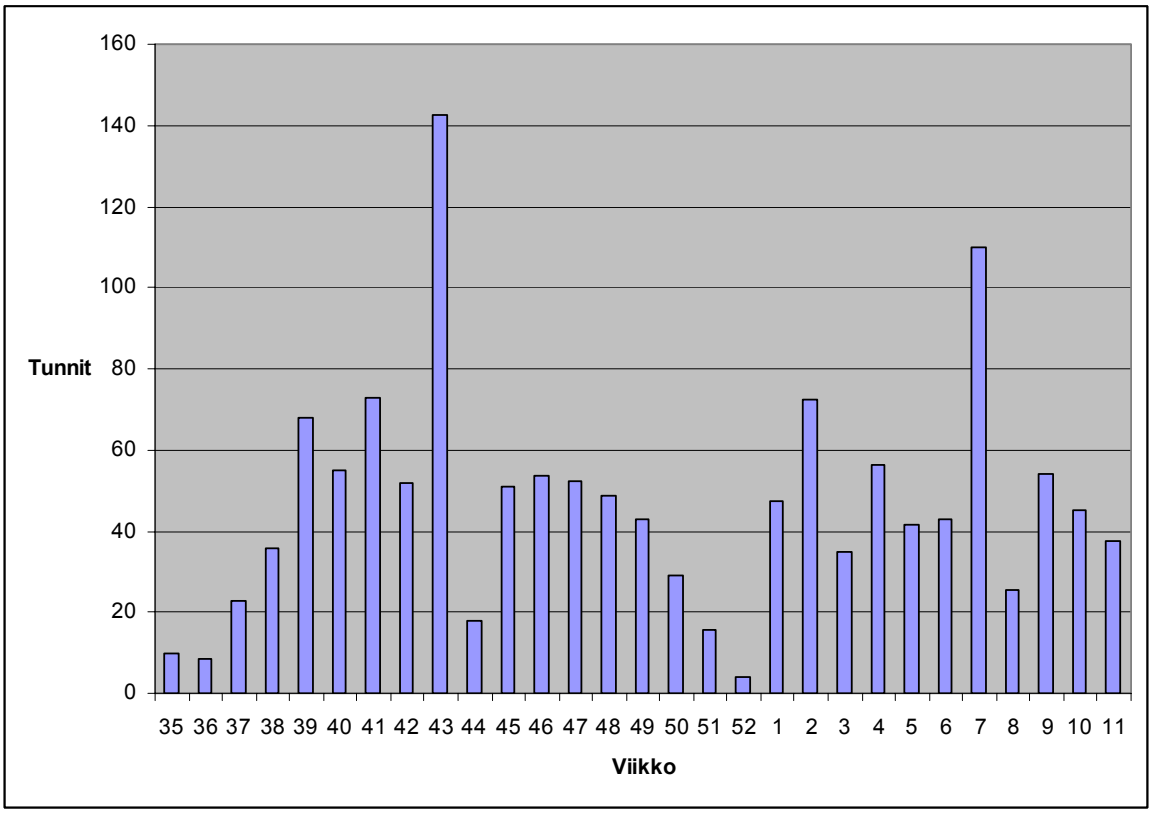
*Taulukko 3: Dokumentit.*

| Proposed | Approved | Implemented | Verified | Deleted |
|---|---|---|---|---|
| 0 | 36 | 0 | 3 | 0 |

*Taulukko 4: Vaatimukset.*

| Kieli | PHP | Javascript | CSS | SQL |
|---|---|---|---|---|
| Rivimäärä | 7426 | 2179 | 430 | 604 |

*Taulukko 5: Koodirivit.*



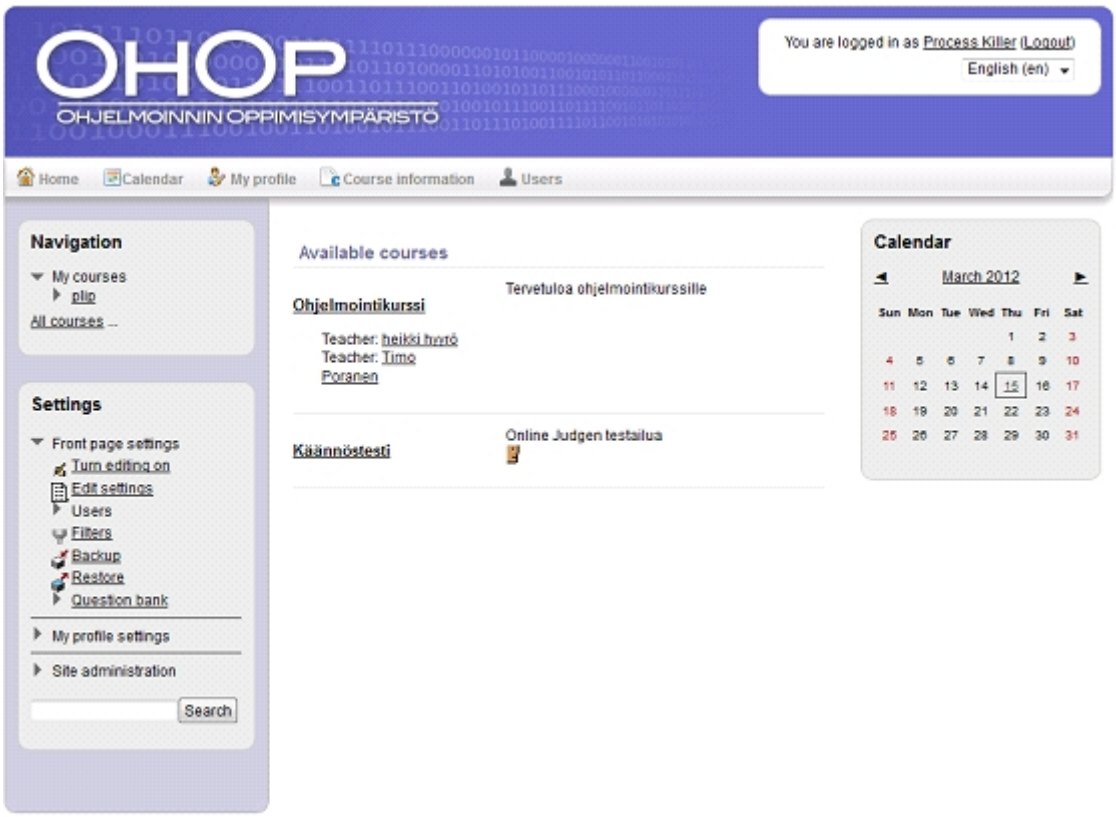

Kuva 2. Projektin viikkotunnit.

# Moodle Project

## Overview

The main goal of the project was to create a learning environment for the School of Information Sciences. The system was implemented as an online-service and powered by Moodle web application platform.

Teachers can make courses and attach coding tasks to them. When a reply to a coding task is sent to the system by a registered user (usually student), it compiles and reviews the reply against given standard output. The user is then given the feedback if her answer is approved. Current supported languages are C and C++.




### Organization and management

### Project managers

Tommi Ritola and Mervi Ollikainen

### Developers

Katariina Tiitinen, (user interfaces)

Risto Salo (user interfaces, Moodle core)

Katja Sorjamaa (documents, usability testing)

Toni Mikkola (database, usability testing)

Ville Valkonen (server, installations, Moodle core)

## Client

Timo Poranen, University of Tampere, School of Information Sciences

Heikki Hyyrö, University of Tampere, School of Information Sciences

## Cource supervisor

Pekka Mäkiaho, University of Tampere

## Methods and tools

- Git (version control)
  - easy and reliable version control tool
- Tortoise Git (version control)
  - Windows tool for managing Git
- XAMPP (Web server)
  - "de facto" web server -solution for Windows+Apache+PHP
- NetBeans IDE 7.0 (development)
  - useful and quite light integrated development environment with good PHP plugins
- Redmine (project management)
  - light web-based project management and bug-tracking tool
  - worked well for our purposes
- Moodle 2.2.1 (framework)
  - an open source web application platform for creating learning environments
  - tight learning curve, incoherent code and documentation
- Onlinejudge (plugin/module)
  - an open source module that extends the core features of Moodle
  - manages the code compiling and reviewing

## Project phases and development model

We chose Scrum as our development model and modified it a bit to meet our more simple needs. For example daily sprints were not possible so we met only once in a week and kept up the conversation by other means.

The project was divided into nine two-week sprints that all started with sprint's start meeting and ended to a review meeting. We also had larger review meetings with the customers and course supervisor once in a month.

| **Scheduled task** | **Start date** | **End date** |
|---|---|---|
| First meeting | | 13.09.11 |
| Preliminary analysis | | 28.10.11 |
| Project plan | | 12.10.11 |
| Personal report I | 01.11.11 | 07.11.11 |
| 1st sprint | 11.10.11 | 25.10.11 |
| 2nd sprint | 25.10.11 | 08.11.11 |
| 3rd Sprint | 08.11.11 | 29.11.11 |
| 4th sprint | 29.11.11 | 13.12.12 |
| 5th Sprint | 13.12.12 | 17.01.12 |
| 6th Sprint | 17.01.12 | 31.01.12 |
| 7th Sprint | 31.01.12 | 14.02.12 |
| 8th Sprint | 14.02.12 | 28.02.12 |
| 9th Sprint | 28.02.12 | 07.03.12 |
| Personal report II | 01.01.12 | 15.01.12 |
| Midterm Presentation | | 30.11.11 |
| Final Report | | 15.03.12 |
| Project Story | | 15.03.12 |
| Review meeting I | | 29.11.11 |
| Review Meeting II | | 13.12.11 |
| Review Meeting III | | 23.01.12 |
| Final Meeting | | 15.03.12 |
| Final presentation | | 14.03.12 |
| Usability testing | | 23.02.12 |

## Experiences

Eight risks were recognized in the beginning of the project and they are listed below:

| **Risk** | **Impact** | **Propability** |
|---|---|---|
| Project member quits the course | 3 | 3 |
| Project member doesn't have enough time because of other commissions | 4 | 4 |
| Technical know-how is not high enough | 4 | 2 |
| Schedule is delayed | 2 | 3 |
| Project member is away for a long time | 2 | 2 |
| Requirements change during the project | 3 | 1 |
| Motivation doesn't last for the whole project | 4 | 2 |
| Poor working atmosphere | 2 | 1 |

1 – low, 2 – medium, 3 – high, 4 – very high

The most significant risk that came true was that most of the project members were working full time during the project. The motivation and the time needed didn't always reach the required levels.

The project group liked that this course gives a real life experience of how to be a part of a larger project organization. Both developer and project leader roles are experienced during the course. The course is often the first course at University of Tampere that gives the possibility to develop software in a group. It brings experiences for example from version control and code reviews that may be a bit hard to get otherwise.

Some project members thought that the course is too long-lasting and could be more compact and last for example two periods. On the other hand it's obvious that software projects do take much of time in real life too.

## Statistics

| Team size | Dev. model | Start date | End date | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | Scrum | 13.09.11 | 15.03.11 | 185 | 1033 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | Design | Code | Integration and testing | Reviews | Repair | Study+ Other | Total |
|---|---|---|---|---|---|---|---|---|---|
| Hours | 236,5 | 42 | 32,5 | 172,5 | 142,5 | 31,5 | 36,5 | 339 | 1033 |
| % | 22 | 4 | 3 | 17 | 14 | 3 | 4 | 33 | 100% |

*Table 2: Group effort by activity.*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 12 | - | - | - | - | - |

*Table 3: Requirements and high-level design outcomes.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| - | - | - | - | - | - |

*Table 4: Design outcomes.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 7 | 4 |
| Project Plan | 26 | 7 |
| Usability analysis | - | - |
| Requirements specification | - | - |
| User privileges and roles | 3 | 1 |
| Scoreboard manual | 6 | 1 |
| Usability test report | 11 | 2 |
| Final report | 14 | 1 |
| Project's story | 5 | 1 |
| Weekly reports | 21 | - |

*Table 5: Documents.*

| Language | PHP | JavaScript | CSS |
|---|---|---|---|
| LOC | 2057 | 358 | 726 |
| SLOC | 1654 | 300 | 636 |
| Reused code | ~100 | - | ~94 |
| Reused and modified | ~599 | - | ~424 |

*Table 6: Codelines.*

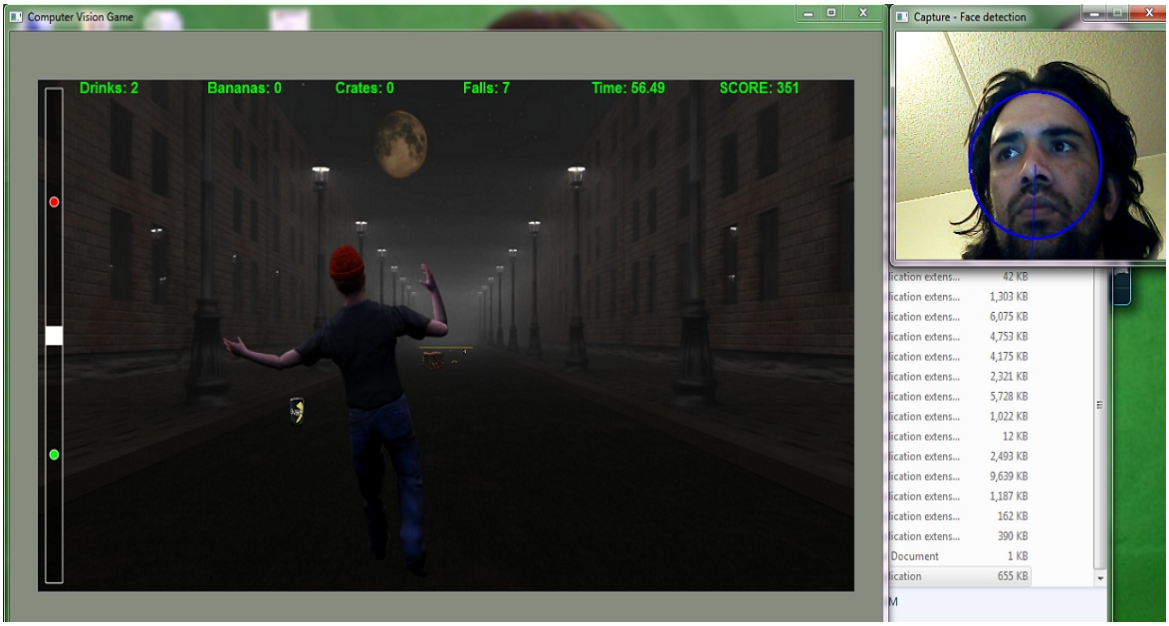
# Computer Vision Game

## Overview

The goal of the project is to create hands free games which replace use of mouse and keyboard by visual gestures for the TUCHI, Department of Computer Science. The targeted users are of any age group who likes to play the game without much more effort. The main objectives of the project which were fulfilled in the project are:

- Design of the game which can be controlled by visual gestures.
- Implementation of computer vision methods for tracking, detection and recognition
- Design of the experiment and carrying out pilot tests.

To fulfil these requirements we developed the game named as **Take Drunkard Home**. The game concept is to have competition between the two drunkard roommates from the different location to reach the home to get a last piece of beer on apartments. The movement of the one drunkard person is controlled through the head movement instead of mouse.




## Organization and management

**Project managers:**

Tek Prasad Gautam

Reza Ahliaraghi

**Developers:**

Mirjan Merruko, Yanzhao Wen, Anju Thapa, Henrik Lehtinen and Esko Vankka

*From left: Reza, Henrik, Anju, Tek, Esko, Mirjan, Yanzhao*

**Client:**
Dr. Yulia Gizatdinova and Jonne Iso-Tuisku

**Course Supervisors:**

Timo Poranen and Pekka Mäkiaho

## Methods and tools

The game client was done by using QtSDK and programming language was C++. Communication between client and server uses HTTP protocol. Server side is using PHP and Symphony framework.

A Google wiki was used for communication, keeping track of schedule, sharing of documents and distributing other information. We use Google code repository for version control system. Hour tracking is done with Google Docs by using form that team members can easily fill. The tools used for documents are Microsoft Word, Open Office. Spreadsheets are maintained by Microsoft Excel and Google Spreadsheets.

## Project phases and development model

Our development model is Scrum with six sprints. Each Sprint was designed for two weeks. By the beginning of each sprint sprint-specific requirements were pushed to

the current sprint backlog. We reviewed each sprint after the two weeks period and if any things remain to do then we move that task in the next sprint. Instead of daily scrum meetings we held similar meetings once or twice a week depending on current situation.

| **Scheduled task** | **start date** | **End date** |
|---|---|---|
| Short presentation | 14.09.2011 | 14.09.2011 |
| Preliminary analysis | 16.09.2011 | 22.09.2011 |
| Preliminary analysis review | 26.09.2011 | 26.09.2011 |
| Project plan | 27.09.2011 | 10.10.2011 |
| Project Plan review | 18.10.2011 | 18.10.2011 |
| Personal report I | 01.11.2011 | 07.11.2011 |
| First sprint | 31.10.2011 | 13.11.2011 |
| Second sprint | 14.11.2011 | 27.11.2011 |
| Third Sprint | 28.11.2011 | 11.12.2011 |
| Fourth sprint | 12.12.2011 | 08.01.2012 |
| Fifth Sprint | 09.01.2012 | 22.01.2012 |
| Sixth Sprint | 23.01.2012 | 05.02.2012 |
| Product backlog | 30.10.2011 | 23.1.2012 |
| Personal report II | 01.01.2012 | 15.01.2012 |
| Midterm Presentation | 30.11.2011 | 30.11.2011 |
| Final Report | 15.02.2012 | 14.03.2012 |
| Project Story | 15.02.2012 | 14.03.2012 |
| Review meeting I | 25.11.2011 | 25.11.2011 |
| Review Meeting II | 13.01.2012 | 13.01.2012 |
| Review Meeting III | 10.02.2012 | 10.02.2012 |
| Final Meeting | 16.03.2012 | 16.03.2012 |
| Final presentation | 14.03.2012 | 14.03.2012 |
| Usability testing | 05.03.2012 | 09.03.2012 |

## Experiences

We liked the possibility to get programming experience the project provided. The project showed us real problems and issues probably also faced in commercial real-life software development and software projects. We got much time management expertise, both personal and for organizing the time management issues for the whole team. Also the project showed ourselves our personal level of expertise.

As we were probably the most multicultural project this year, we don't have to face communication problems due to cultural and language differences. Being a cross-cultural team also proved to be an advantage. It forced us to improve our understanding of different people and to share our cultural values with each other, thus the team work taught us more than otherwise possible. The project helped us to improve our interpersonal skills. In the future we will be better off when working in cross-cultural projects. Also in brainstorming new ideas the different backgrounds we had proved to be an advantage.

## Statistics

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | Scrum | 01.09.2011 | 11.03.2012 | 190 | 1594 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 240 | 97,5 | 169 | 247 | 131 | 76 | 28 | 393 | 209,5 | 1594 |
| % | 15,05 | 6,11 | 10,60 | 15,49 | 8,21 | 4,76 | 1,75 | 24,65 | 13,14 | 100% |
| Usabi-lity | - | - | - | - | - | - | - | - | - | - |
| Total | 240 | 97,5 | 169 | 247 | 131 | 76 | 28 | 393 | 209,5 | **1594** |

*Table 2: Group effort by activity.*

| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| 30 | - | - | - | - | - |

*Table 3: Requirements and high-level design outcomes.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| - | - | - | - | - | - |

*Table 4: Design outcomes.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 8 | 2 |
| Project Plan | 19 | 2 |
| Usability analysis | - | - |
| Requirements specification | 4 | 1 |
| Design plan | - | - |
| User interface document | 16 | 1 |
| Test plan | 1 | 1 |
| Test report | - | - |
| Usability test report | 1 | 1 |
| Final report | 17 | 1 |
| Project's story | 6 | 1 |
| Weekly reports | 28 | 1 |
| Implementation Plan | 1 | 1 |
| Inspection and review | 3 | 1 |
| | | |

*Table 5: Documents.*

| | |
|---|---|
| Language | C++ |
| LOC | 3912 |
| SLOC | 3912 |
| Reused code | opencv (open source), haarcascade.h, haarcascade.cpp, haar_face_cascade.xml, haarcascade_profileface.xml (from client) |
| Reused and modified | postprocessing.cpp |
| Classes | 24 |
| Functions | 230 |
| Code revisions | - |

*Table 6: Codelines.*

# Snakes and Ladders for Windows Phone 7

## Overview

The aim of the project was to develop multiplayer version of classic Snake and Ladders game for Windows Phone 7 that can be played against other players on multiple devices via WI-FI. Another big aim was to learn technologies used for Windows Phone 7 -development.


**Picture 3: Single player game against two computer opponents.**

## Organization and management

- Project managers: Timo Korhonen & Aleksi Tiensuu
- Project workers: Da Ke, He Ye, Markus Leinonen, Sundar Kunwar & Trong Nghia Nguyen

- Client: Irfan Javed, Teleca Finland Oy
- We also had external (not from Teleca) graphics person, Timo Muranen, for our disposal at the beginning of the project that made most of the graphics used in the game.

## Methods and tools

- Visual Studio 2010 with Windows Phone 7 Mango SDK; C# (Silverlight & XNA).
    - Windows Phone 7 emulator.
    - Windows Phone 7 developer activated version provided by Teleca and another by Microsoft Student Tech Club Tampere.
- MS Office 2007 & Open Office
- Redmine project management platform (wiki, issue tracker, time logging etc.).
- Subversion version control.
- Skype.
- Design principles: agile (from abstract to specific and through crude implementation to polished end-result).
- Development model: ABC-sprints.

## Project phases and development model

We used development model called "ABC-Sprints" *(ABC-Sprints: adapting Scrum to academic game development courses, by Jonas Schild, Robert Walter and Maic Masuch)*, that is a modification of Scrum for academic game development courses. We modified it a little to make it better suit for our needs and take in count our limitations. The idea was that in Alpha-sprint prove of concept would be made, something that indicates that our approach to develop the game works. Goal of the Beta-sprint was to result feature full version of the game, that would have been polished in the Completion-sprint. Instead of daily scrum-meetings, we had a weekly meeting and at some point, around middle of project, we introduced a weekly coding session to boost the development process.

Our preliminary schedule for the releases was that Alpha-sprint would have been released on week 45 (year 2011), Beta-sprint 6 weeks after that and Completion-sprint in 7$^{th}$ week of 2012. We scheduled 2 weeks flex - aimed ending date was set earlier than required - so that if we can't keep up with the schedule, it will not automatically result trouble. Existence of the scheduled flex was only informed to project supervisor Pekka Mäkiaho, not to project members, so that it would better serve its purpose.

Sprint finished around the time we held reviews: Alpha-sprint review 23.11.2011, Beta in 12.2.2012 and Completion in 28.2.2012. We finished Alpha-sprint in the schedule and we managed to do the "proof of concept", but then we encountered some problems in Beta-sprint and we had to use our reserved time and little bit more for it. Completion-sprint started late and lasted little bit shorter time that we planned, and it was not the polishing-sprint we originally planned it to be. It was more like what we had planned for the Beta-sprint.

# Experiences

## Risks

We actually managed to figure out most of the risks at the beginning of the project. However we could have done more to prevent them from becoming reality. The two risks that came almost totally from behind a tree were: At the very beginning, the initial expression was that our team is way more skilled than it actually was and the lack of development tools, as the development environment set relatively high requirements for the hardware. We managed to acquire more hardware as people got new computers and we received a WP7 devices in the end though.

## Experiences

The culturally diverse workplace may never be completely free of communication problem; however a lot of potential solutions of the problem were taken into consideration such as using pen and paper, marker and board etc. for effective and efficient communication.

At the beginning the approach to the project was overly optimistic and the Alpha-sprint contained way too many features to be completed with the group at hands. The group had few more advanced developers and 3 novices. Because of that, the workload for the more experienced developers was heavier that we would have liked it to be. Alpha-sprint resulted what it was supposed to result, but Beta-sprint was kind of a failure and most of the things that were supposed to achieve in the Beta-sprint were done in Completion-sprint. Due to that there was not enough time to polish the game as much as we'd liked to.

We had some problems to get equipment that actually can run the development environment and it slowed us down at the beginning and later on, we would have had use for devices (phones) when we did not have access to them.

Working with a multicultural team and working with Windows Phone 7 were new for everyone, so everyone got some new experiences on both and learned new things, which is an important part of a university project. And while everything didn't always go as planned, we think that the project at least taught a lot about teamwork and of course Windows Phone 7 development.

Working hours used for the project produced very different outputs; some members were meaningfully productive while others needed more time for studying etc.
Project finished in time with all the required features, however not without difficulties and not without bugs. Overall the project was a success, because we got the game ready in the end and people learned new technologies.

## "Things to do better next time"

- When the programming environment and language is new, there should be more systematical plan on how to get started with it, how to study and learn.
- Project managers should better estimate the project time and the project difficulties according to project members' working experiences.

- There should be more coding sessions where every members can exchange working experience as well as work together so that would help the project progress to be improved faster.
- There should be a commitment for every members in adopting working standard such as standard for working with code repository, code standard ...
- There should be more supports from university and project clients in order to have better working environment and hardware devices relating to the project.
- There was the situation that there were only developers but no graphic designer in the project group. Therefore, it would be better if the course supervisor can increase the diversity in working skills in a project group.
- Communication problems need to be solved as soon as they appear or even before.
- Much more time should be used to study alternative technologies and ways to do get things done before choosing a path to follow.
- Strengths of each member should be explored in more detail level to get a right person working with a right task.
- Splitting work to really small tasks might help to balance workload on a varying skilled group.
- Culture is the centre of functioning. Multicultural issues such as working style, communicating style etc. needs to assertive measure before setting group.

## Statistics

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | Agile | ~ 8.9.2011 | 16.3.2012 | 191 | 1519,5 |

*Table 1: General project information.*

| Activity | Planning and management | Req. specification. | De-sign | Code | Integration and testing | Reviews | Repair | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 398 | 13 | 82 | 391 | 26 | 19,5 | 15,5 | 303 | 271,5 | 1519,5 |
| % | 26,2 | 0,9 | 5,4 | 25,7 | 1,7 | 1,3 | 1 | 20 | 17,9 | 100% |

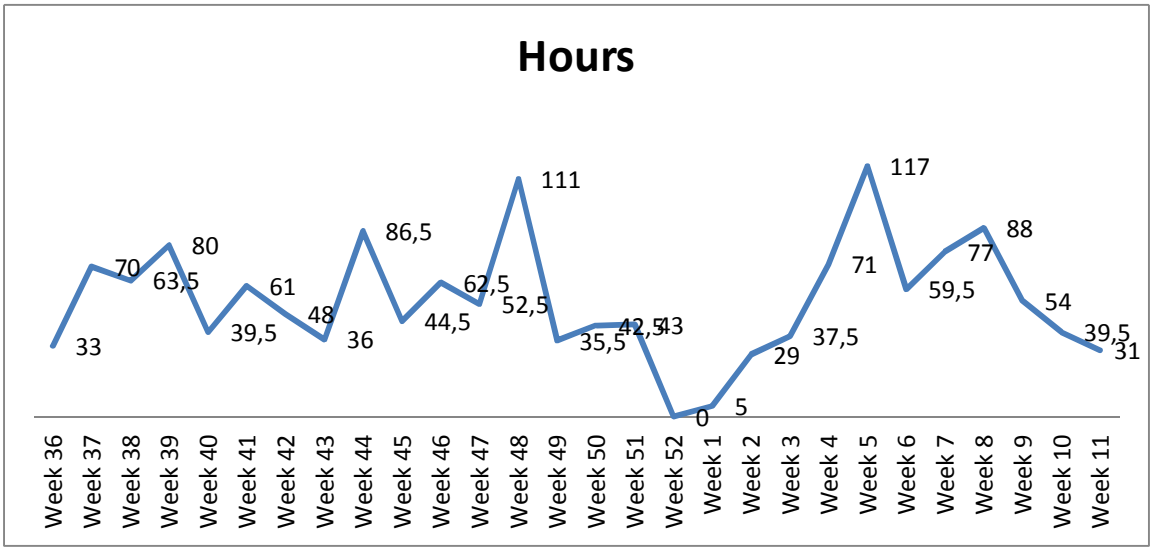
*Table 2: Group effort by activity.*

| Language | C# |
|---|---|
| LOC | 12152 |
| SLOC | 8812 |
| Code revisions | - |

*Table 3: Code lines.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 6 | 1 |
| Project Plan | 13 | 3 |
| Usability analysis | 4+2 | - |
| Final report | 15 | 5 |
| Technical documentation | 13 | 4 |
| Project's story | 6 | 2 |

*Table 4: Documents.*




*Picture 4: Weekly working hours.*

# Customizable Process Visualization

## Overview

The project goal was to implement a tool for easily creating interactive payment-related visualizations. The client company already had a few such visualizations, but they were tedious (and as such, expensive) to make because each was a separate program.

So we set out to create a flexible solution that could be used to create these presentations without any programming involved from the creator's part. Because our presentations were saved in a custom file format, we also wrote a player that could show the created visualizations in a web-browser. Later on the goal of strictly payment-related visualizations was expanded to suite other, more generic kinds of presentations as well.

## The program

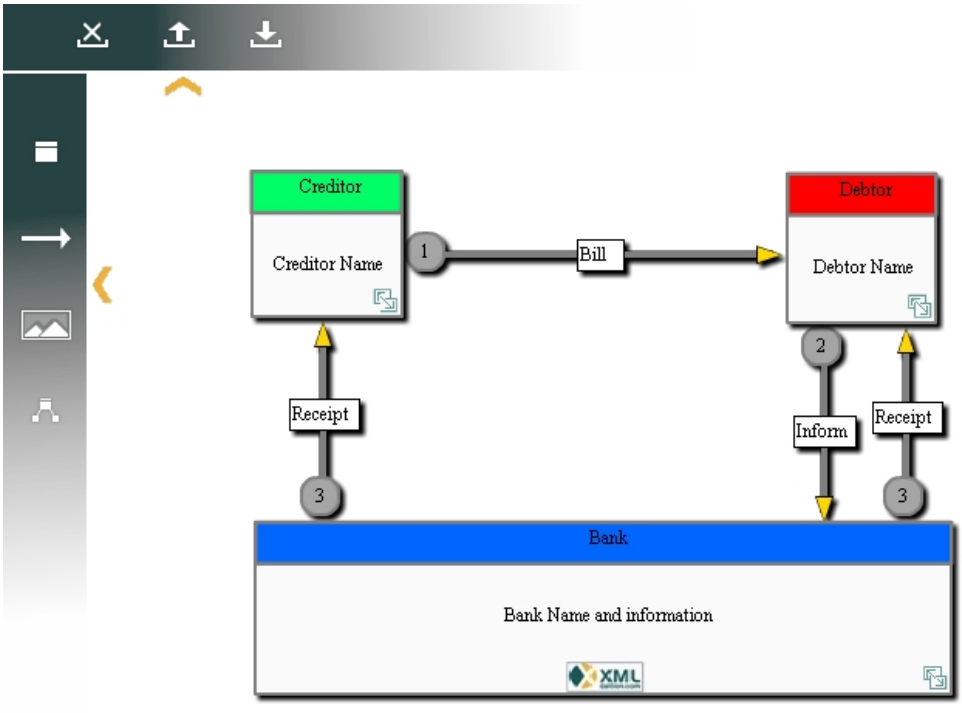



**Figure 1:** General view of the edit mode.

### Edit mode

Edit mode allows the creation and editing of existing presentations. As the knowledge of the user's using this mode is at expert level (users will be the client's own staff), we focused on enabling the user to be able to create presentations efficiently, but trying not to forget general usability.

**Figure 2**: The menu bar

Features:

- **New button** can be used to open a new window and current unsaved content will be lost.
- **Open button** opens a second window that lets the user choose the file which he/she wants to open.
- **Save button** is to save the current presentation (also opens a file picker).

**Figure 3:** The toolbar

Features:

- **Nodes**: With "node" in this document (and in the program), we refer to a rectangular shape in the presentation, representing some sort of entity, which can send or receive information through connectors. Placing a node is done by clicking the node button in the toolbar and then clicking on canvas at desired spot.
- Double-clicking a placed node opens edit window of node, through which user can edit the text, color and other properties. The nodes can be moved by dragging them, and their sizes can be changed by dragging the icon at the right corner of nodes.

- **Connectors**: Lines connecting nodes, representing information transfer. To add one, user clicks connector button (arrow) in the toolbar, and then click a "where" node and finally click another, "to" node. Program will try to automatically place an arrow between them. Because the algorithm to place the connector is not always very smart, user can also drag the joints in the connector to make it follow the desired route should the automation fail to give satisfactory results. If the nodes are moved, attached connectors will be automatically adjusted as well. Like with nodes, double clicking the title or the order number of a connector will open a edit window to change attributes of connectors (such as title, info, colour etc.).
- **Picture:** Pictures can be added to the presentation by clicking picture button in the toolbar and selecting the appropriate file from a window that will open up. The pictures can be moved and resized (behold, also deleted as of latest update).
- **Option Box:** Option box is used to change the data flow of visualization. I.e to create alternative paths for the flow depending on user's actions. Optionboxes contains buttons that, when pressed, trigger other actions.

### Presentation mode

Presentation mode is the mode visible for public, that plays the saved presentations.

**Figure 4:** Menu bar for the presentation mode

Features:

- **Open Button:** When user opens a saved file, the whole scenario of information visualization of data including all nodes and flows will be shown up in the translucent way.
- **Go to start:** It can reset the whole process of presentation.
- **Play Button:** While user presses play button, the data flow will begin and one by one to show up to let the user know the process of data flow. In this process, play button becomes pause button which can be used to pause.
- **Fast Forward button:** Skips to next step without animation
- **Go to End:** Skips to end of the presentation
- **Stop button:** It will make the flow process into the completed scenario of information visualization of data, which means the flow diagram with all flows and order numbers.

## Organization and management

*Project team. From left right: Otso, Bo, Li, Qin, Mika, Antero, Jussi*

| Name | Role |
|---|---|
| Antero Mäenpää | Manager |
| Jussi Ampuja | Manager |
| Otso Leppälä | Developer |
| Qin Yameng | Developer |
| Bo Huang | Developer |
| Zhenxing Li | Usability expert / tester |
| Mika Suokas | Usability expert / tester |

## Tools

Adobe Flash CS (versions 4 to 5) - chosen as IDE, (inherently, ActionScript3 became the programming language). Interestingly heavy, but still lacks a lot of functionality one would expect from integrated development environment. Commercial - some team members had to settle for trial version.

Adobe Photoshop - Coupled with pen and paper quite an effective way to design UI related stuff.

Mediawiki - One of the most basic wiki-platforms. Gets the work done, offers everything that is needed but not much more.

## Project phases and development model

## Development model

Project management started out with somewhat basic scrum, with weekly meetings, 3-week iterations and project/sprint backlogs (no designated scrum-master though). It continued that way till the end, but the focus was shifted from knowingly applying development model into something more free-form.

In the last months it would be more realistic to say that we were focusing more on one week iterations, where one day was used for status checking and idea sharing, and possibly other day for workshopping online / offline, in addition to independent working. Weekly meetings were held every week the project was active. Due to requests from team's members we had to give in and not have the meeting at Christmas day.

Due to the official person designed for acting as a client from XMLdation was often busy, Antero, also working in XMLdation, took some of the client's responsibility.

Several workshops were held during the week in addition to the weekly meeting. One of these was replaced with an online workshop as team members at that time were scattered thorough Europe.

### Milestones

Project plan inspection, 1st review meeting 4.10.2011

2nd review meeting
3rd review meeting
Final meeting

| **Milestone** | **Date** |
|---|---|
| Project plan inspection | 4.10.2011 |
| 1st review meeting | 17.11.2011 |
| 2nd review meeting | 20.12.2011 |
| 3rd review meeting | 23.1.2012 |
| Final meeting | 22.3.2012 |

**Agenda for meetings:**

**Preliminary Analysis**

Short presentation about the client
Short info about the project
Going through the preliminary analysis paper
Demola Kickoff reminder

**1st review meeting**

*'''Official meeting'''*
Reviewing sprint and project status
Analysing project plan
*'''Weekly meeting'''*
*Scrum summary
*Welcoming new member!
*More thorough look into completed sketches
*New ideas
*Team ratio currently is 3/2 UI/Programming. Discussion to make this 2/3
*Weekly meeting time change
*Usability analysis
*Dividing weeks tasks for everyone

**2nd review meeting**

Status presentation
Ideas

*Notes*

Good feedback was gain from the meeting.

Now we can start shifting the focus from technical side to testing and general look and feel from the actual user point-of-view. While keeping in mind that we should be able to implement a solution which can match visualization features of the previous version.

Antti Salomaa informed us that Demola will be hosting testing events on thursday, which we can take part of as well.

**3rd review meeting**

Status presentation
Ideas

**Final meeting:**

Project status
Presenting Edit mode
Presenting View mode
Further Ideas
Possibility for acquiring currency

## Experiences

One of the listed risks speculated in the beginning was language barriers and other culture-related misconceptions, but that risk was not met to full extend. Some problems did occur, but nothing fatal. It goes without saying that communication in general is one of the most important parts in projects.

Freezing feature list and focusing on fixing existing bugs instead of adding new features is important to do early enough. It's easy to end up in a situation where you have myriad implemented features which are not quite ready, but due to deadlines adding new features is still kept on a high priority.

Physical workshops where people work in a shared space with the possibility for face-to-face collaboration can be great for fixing numerous of minor bugs or adding small features. On harder and more time consuming tasks people seem to prefer working on their own time. Taking advantage of these facts has the potential to be very useful.

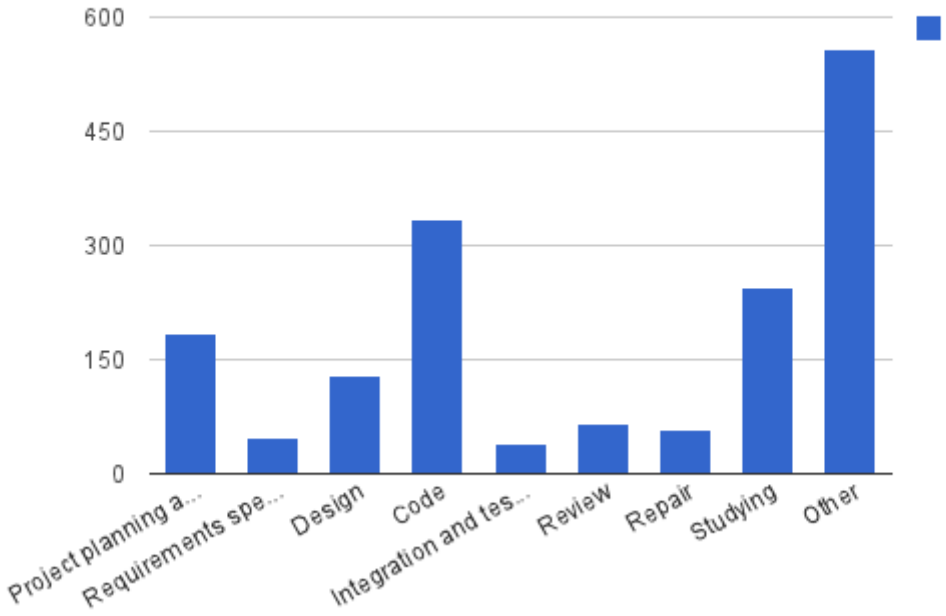
## Statistics

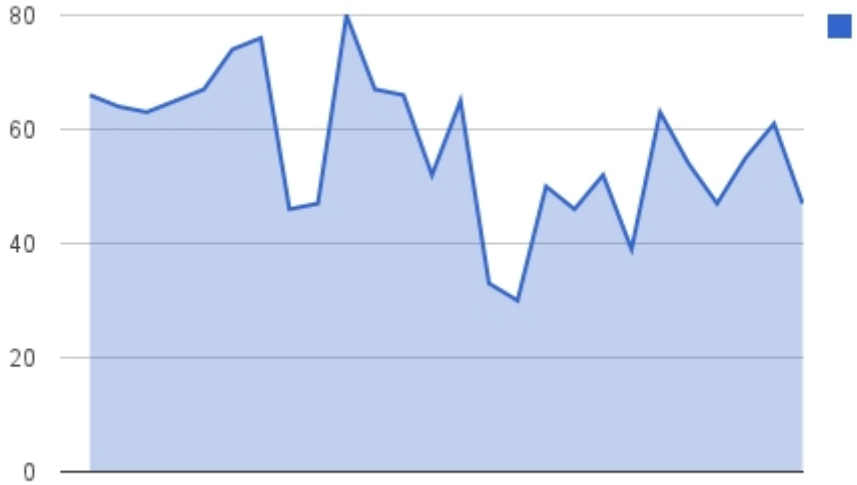
| Team size | Dev. model | Start date | End date | Days | Hours |
|---|---|---|---|---|---|
| 5+2 | SCRUM/some ad-hoc method | 12.9 | 6.3.2012 | | 1662 |

*Table 1: General project information.*

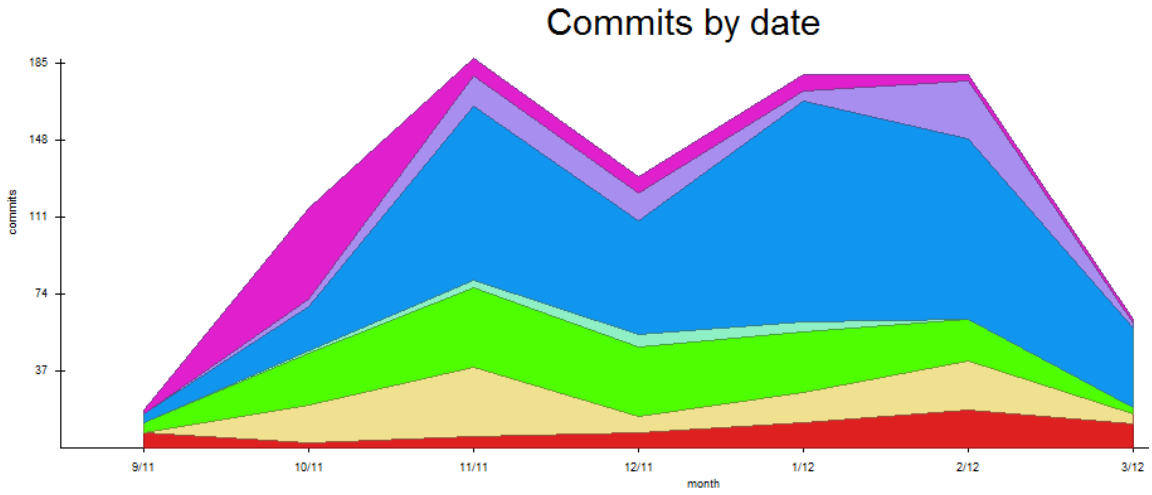
| ***Activity*** | **Hours** | **Index** |
|---|---|---|
| *Project planning and management* | *183* | *0.110* |
| *Requirements specification* | *47* | *0.028* |
| *Design* | *130* | *0.078* |
| *Code* | *333* | *0.200* |
| *Integration and testing* | *40* | *0.024* |
| *Review* | *67* | *0.040* |
| *Repair* | *58* | *0.034* |
| *Studying* | *245* | *0.147* |
| *Other* | *559* | *0.336* |
| ***Total*** | 1662 | *1* |

*Table 2: Group effort by activity.*


600
450
300
150
0
Project planning a...
Requirements spe...
Design
Code
Integration and tes...
Review
Repair
Studying
Other


80
60
40
20
0

*Hour amount*




| Number of requirements | Pages | Use-cases | UI screens | Database diagrams | Database tables |
|---|---|---|---|---|---|
| | | 2 | 11 | 0 | 0 |

*Table 3: Requirements and high-level design outcomes.*

| Pages | Overview diagrams | Class diagrams | Sequence diagrams | State diagrams | Other diagrams |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 0 | 2 |

*Table 4: Design outcomes.*

| Language | ActionScript3, HTML |
|---|---|
| LOC | 7966 |
| SLOC | can not possibly know |
| Reused code | - |
| Reused and modified | 12 (lines) |
| Classes | 41 |
| Functions | 411 |
| Code revisions | 864 |

*Table 5: Codelines.*

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 4 | 1 |
| Project Plan | 13 | 2 |
| Usability analysis | - | - |
| Requirements specification | 0 | 0 |
| Design plan | 0 | 0 |
| User interface document | - | 2 |
| Test plan | 3 | 1 |
| Test report | 4 | 1 |
| Usability test report | 4 | 1 |
| Final report | ? | 1 |
| Project's story | 13 | 1 |
| Weekly reports | ~1 / report | 25 |

*Table 6: Documents.*

# Category Browser

## Overview

The goal of the Category Browser project was to innovate and implement an efficient and usable web-based data browsing solution for Nokia Siemens Networks. The project was a Demola project and was managed at the Demola premises. The client was looking for an application solution that could be place on the company server and used by Nokia Siemens Networks personnel and customers. These customers may also have many categorized needs and requirements towards NSN's products and services. As a whole, there is huge amount of data inter linked that need to be exposed in categorized form. The purpose of this project is to design an information browser that can expose this data fog with clear categorization filtering.

This project is a new concept developed by Atte Karvinen and our group is starting from scratch based on his concept. The client wants to host the project in their Intranet and the target users are NSN's employees.

The client had had some previous similar project to show and some specifics that the application should have both from the interface side and the data-structure side, but otherwise the team got quite free hands and a challenge to come up with some new and more innovative solution. Used technologies were chosen by the team based on previous experiences and abilities. The application was done by using Java in Net-beans development environment and the data-base side was implemented by using MySQL .

The challenge in the project was how to show visually the relations between the data base instances as this was very important for the client. The team managed to create a working idea of the application and after a few interface mockups the programmers started their work and didn´t take too long for the first prototype to be ready and from the start the project kept quite perfectly on schedule and the development process was stable.

Application was primarily developed to work in Internet explorer 8 and later also in all the other major browsers also. The first working release of the software was installed into the clients server for him to evaluate it.

## Organization and management

The team is a multicultural team with team members from Finland, Nepal, Pakistan, and China. It consists of two project managers and five development members. The team had two project managers, Wuping Yao and Sunil Chaudhary. The team used feature driven development model. The project was developed one function at a time. The team met twice a week, from which the other meeting was always compulsory and at same time while the other had more flexibility. Also team members met each other's frequently to work together on the features currently on development. The managers were always available and also participated in the development phase.

**The team is:**

- Sunil Chaudhary

  Role in Team: Project Manager
- Wuping Yao

  Role in Team: Project Manager
- Petri Puustinen

  Role in Team: Database Architect
- Xie Liwei

  Role in Team: Web Development
- Mohammad Ejaz Uddin Syed

  Role in Team: Web Development
- Zhang Yi

  Role in Team:Web Development
- Mikko Myllylä

  Role in Team: Designer, Usability and Testing

**Other stakeholders were:**

- Bernard Garvey, Collaborates Internationally, Facilitator, Demola
- Timo Poranen and Pekka Mäkiaho, Course Lecturer and project supervisor

Category Browser Team: (From left) Petri Puustinen, Mikko Myllylä, Zhang Yi, Sunil Chaudhary, Wuping Yao, Mohammad Ejaz Uddin Syed , Xie Liwei.

## Methods and tools

Tools used for Project Management were:

- Redmine (Project Management)
- Project wiki (Project information)
- Google doc (Hour reporting)
- Doodle (Meeting time scheduling)
- Email ,Mobile, Skype (Communication and reporting)

Tools used for design planning:

- Balsamiq
- Axure

Tool for version control was:

- Tortoise SVN 1.7.5

Tools used for Development were:

- NetBeans IDE 7.0.1
- MySQL 5.5 (Database)

Tools used for graphical Development were:

- Photoshop cs5

Others:

- Microsoft Office, PDF Converter, Notepad (Documentation, Presentation Slides, Weekly report)

## Project phases and development model

### *Development model*

Feature Driven Development (FDD) that is a form of agile methodology was used for development. The five activities followed during it are:

- **Develop overall model**

  Walkthrough of the system in the team was done. Then detail walkthrough was done for each domain after serious analysis and discussion the team. All domains were merged into overall model.
- **Build Features List**

  Using the knowledge gained during modeling was implemented to identify the features list. Every feature was decided with as few communication channels as possible.
- **Plan by Feature**

  After discussion in the team, development plan for feature implementation was produced.

- **Design by Feature**

  Features were prioritized depending on their execution in the project. Then task of the features were divided in the team.
- **Build by Feature**

  After design and allocation of task the team worked in implementation. The team strictly met the deadline for the feature implementation except some special conditions. Then code was tested by the developer himself and manager did final inspection. After final build the code was integrated to the main thread.

## *Project phases*

Project started with the client´s idea which was followed by some innovative planning and design processes. After the techniques to be used were decided the programmers started to code the database structure and after that moved to the interface coding. Graphical design and feature planning was a constant process throughout the project. The project processes were also documented and when the application was getting near to its final form also testing was conducted.

### *List of Development phases*

1 Client´s idea
2 Innovating and planning
3 Mockups and design
4 Deciding techniques and studying them
5 Graphical design
6 Coding and prototyping the database part
7 Coding the interface part
8 Pitching and presenting
9 Installing the application to client´s server
10 Documenting
11 Testing
12 Usability testing

### *Group meetings*

In the early phase of the project, the team was having once a week face to face meeting called “Weekly meeting”. The meetings were arranged to discuss ideas and plans. At the end of this meeting, each team members get assigned some tasks for week. This routine continued until the start of project implementation.

When implementation started, the team decided to have twice a week face to face meetings. One day was allocated for ideas and plans discussion as previous while other was allocated for coding session. Coding session was arranged so that team can interact and work together.

During the last three months of the project, the team started twice a week coding session. Since, the team already had prepared the design and each team members were almost clear about the product, the team felt to devote more time on implementation.

***Weekly reports***

Weekly report was sent once a week. It was a continuous medium to report about progress of project to the supervisors. Definitely there were other medium like review meetings and mid-term presentation for reporting; however, weekly report was also about recording and reporting the number of hours each team members have spent for project and their contribution in that time.

***Inspections and reviews***

***Supervisor***

- There was weekly reporting about the progress of the reports to supervisors.
- Post meeting memo was sent after every meeting with client and supervisor.
- A review meeting with supervisor was arranged once a month. During it the team had to do the briefing of the project progress and problems (if any) to project supervisor. At the end the team was receiving important and helpful suggestions from the project supervisor that they always tried to utilize as suited.
- Email was often used by the team to contact the supervisor whenever they felt the need.

***Client***

- The review meeting with the clients was arranged at least once a month and some time twice a month as needed. During this meeting the team had to do the briefing of the project progress and demonstrate the implementation. After which the client was providing his suggestions and correcting the team when there was any deviation of implementation from requirement. These meetings were also medium for the client to experience the implementation and ask for modification or addition in the requirements.
- Email was used to communicate with the client.

***Facilitator***

- All team activities were performed at Demola. So there was always some brief meeting with the facilitator from Demola. During which the team was reporting him about project progress and ask for any resources or help that the team expect from him.
- Email was also used for communication.

Reviews and inspections meeting were arranged in the forms below:

- Client and team
- Supervisor and team
- Facilitator and team
- Facilitator ,client, and team
- Supervisor, client, team
- Supervisor, client , facilitator, and team

There were 6 meetings with supervisor, 7 meetings with client, and many meetings with facilitator. (Post meeting memos were sent to supervisor after every meeting with either client or supervisor.)

Presentations made during project:

- Short project presentation
- Project progress presentation
- Project final presentation

Presentation made in Demola:

- Project progress pitching
- Project Final pitching

**Group work related issues**

Except the weekly meeting, we always have another meeting, or say coding session. Normally, the coding session last quite long, always from 2pm to 10pm. During the session, the managers and developers work together in coding or testing. The communication become easier since everybody is around us. Some of the members prefer to get together to have some food after a whole day's working. The communication chance out of work improves the relationships among these members. The only issue in our group is that some members had some trouble in participating and thus cannot integrate into the team later on. And the managers tried to find some task which is available for these members to do to earn their working hours and contribute to the project.

## Experiences

The project is a good exercise in applying agile development. It is the first time to use feature driven developing method for most of us. To join an international team also brings us a lot of fun. We learned not only the techniques about our major, but also some custom and cultural-related knowledge.

## *Foreseen risks*

### *People leave the project or do not contribute as expected*

***Cause***

- A team member (Project manager and Development members) was unable to manage sufficient time for the project due to his busy schedules and other personal reasons.

***Risk Management***

- The project is carried for credits. It does not involve direct monetary benefits. Thus, only measure to avoid this risk was motivating team to continue project and suggested them to make enough free time for the project.

### *Insufficient knowledge of products and requirements*

***Cause***

- A requirement mention in the client document was understood otherwise due to duality meaning of language used.

***Management***

- It was compulsion for every team members to be present in the meeting with client.
- In case of confusion, the client was asked to clear the requirement.
- Implementation of requirements was performed only after discussing it in the team and ensuring that everybody understood something.

***Technology risk***

***Cause***

- All team members had not common platform experience.

***Risk Measurement***

- Technology that is comfortable to majority team members was selected after deep discussion.
- Team members were assigned the task to evaluate tools before giving their view.
- Development session was organized at Demola premise so that team members can help to each other in difficulties and learn from each other.

***Delay to get resources***

***Cause***

- The client was unable to send the dummy data for implementation.

***Risk Management***

- Tested on self-designed probable data.
- Contacted the client to remind him to send data as soon as possible.
- People leave or do not contribute as expected.

## *Risks not foreseen*

### *Working hour and output does not match*

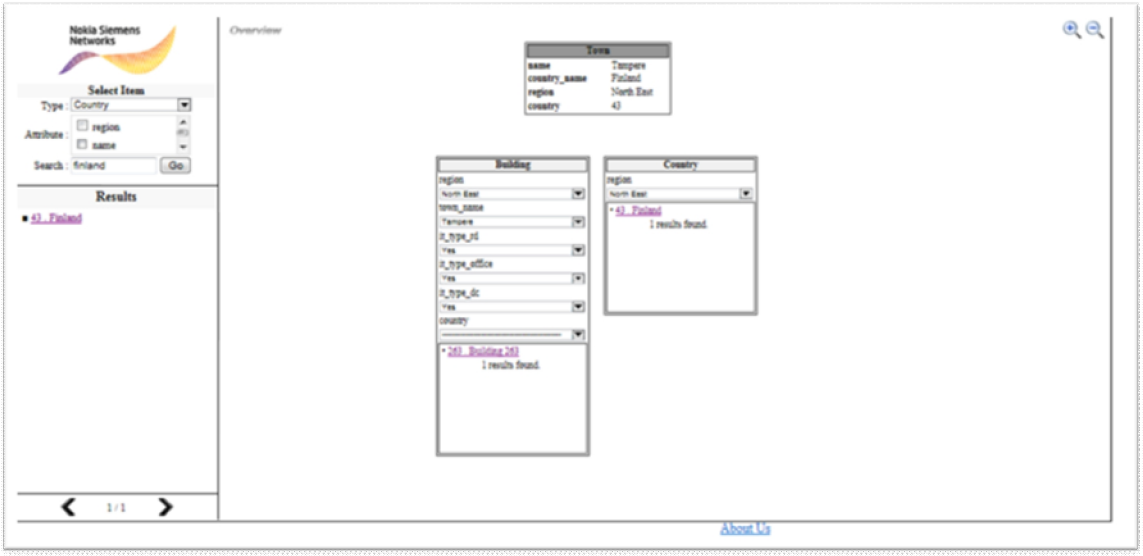
Working hours submitted by some team member and his output was mismatching.

### *Project time exceeded the estimation*

The project time exceeded the estimated time. This could be due to additional requirement forwarded by the client in the mid of the project.

## Screenshot

The Category Browser application in use case: user has searched by item type: country for "Finland".



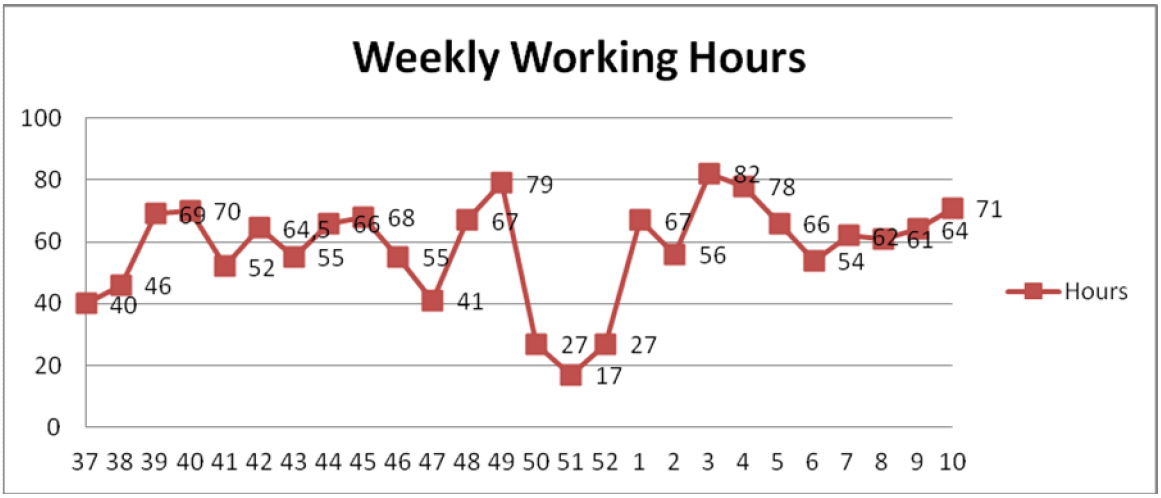
## Statistics

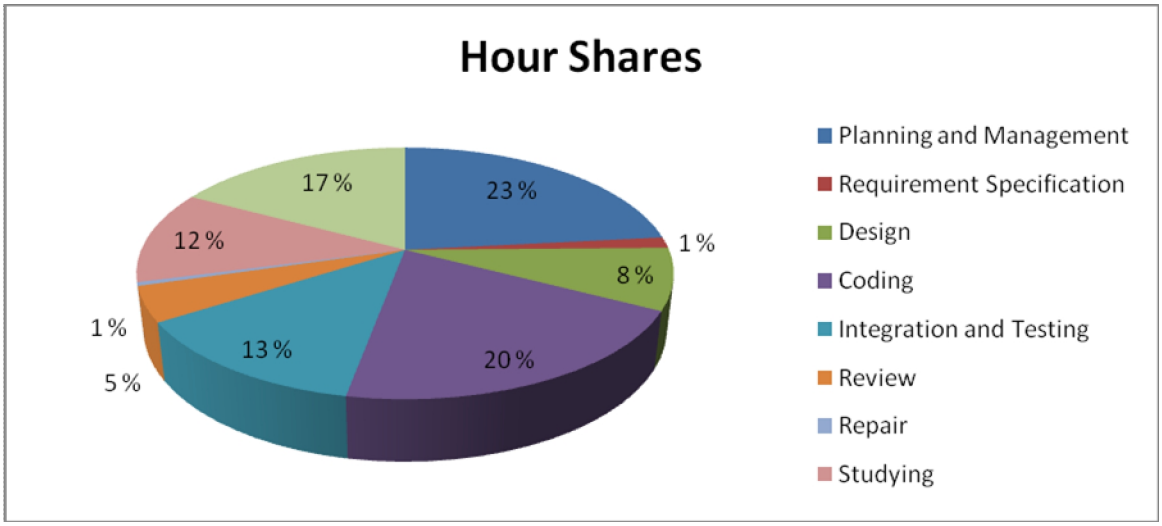
| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+5 | Feature-Driven | 10.09.2011 | 15.03.2012 | 187 | 1501.5 |

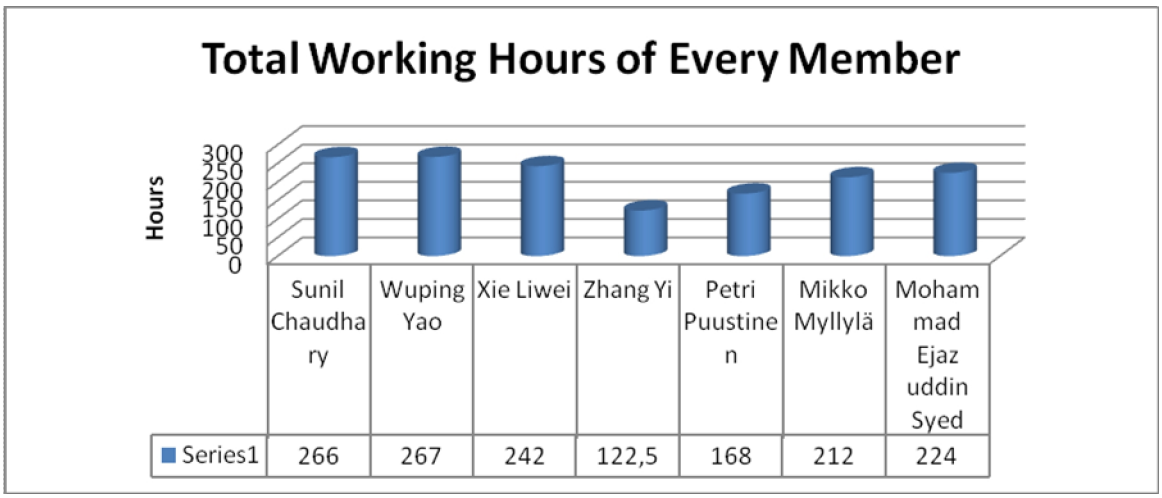
*Table 1: General project information.*

| | |
|---|---|
| Language | JAVA, HTML, JS, CSS |
| LOC | 2945 |
| SLOC | 2288 |
| Reused code | 144 |
| Reused and modified | 144 |
| Classes | 25 |
| Functions | 172 |
| Code revisions | 93 |

*Table 2: Codelines.*



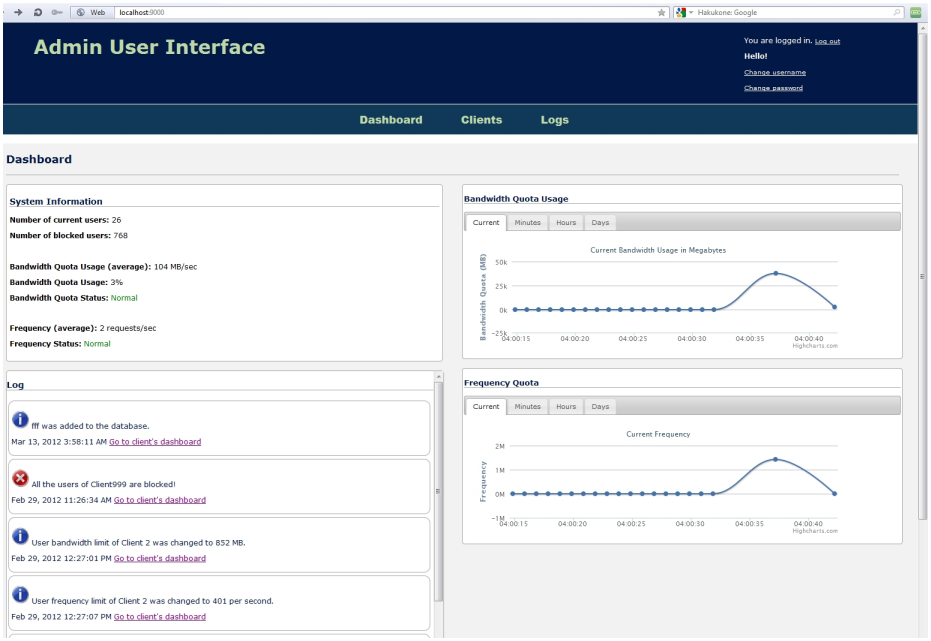

Hour Shares
23 %
1 %
8 %
20 %
13 %
5 %
1 %
12 %
17 %
Planning and Management
Requirement Specification
Design
Coding
Integration and Testing
Review
Repair
Studying


Total Working Hours of Every Member
Hours
300
250
200
150
100
50
0
Sunil Chaudhary
Wuping Yao
Xie Liwei
Zhang Yi
Petri Puustinen
Mikko Myllylä
Mohammad Ejaz uddin Syed
Series1
266
267
242
122,5
168
212
224

# Service load throttling and monitoring for CoWS

## Overview

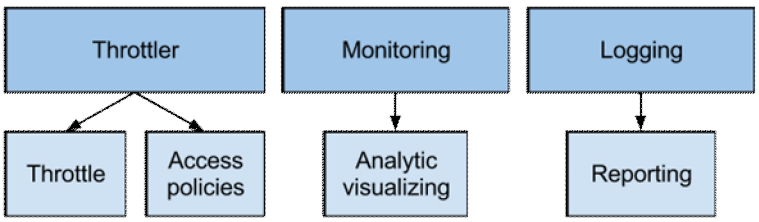
Purpose of the project was to design and implement a Common Web Service throttler and monitoring gateway tool. Cows Admin UI is part of the end-product being the user interface and facade for the managing the Gateway.This product is more like a research product of how a throttler can be implemented in one way. Further development is conducted by Nokia.




The main function was throttling feature which is the back-end component. From the facade the Monitoring and Logging features can be handled and also the access policies.

Throttler's purpose is to do a sort of load balancing for the requests from clients to the web service and limit the superabundant use. Throttler uses policies to determine the tightness of the gateway. One policy includes one or more criteria that defines the throttling rules e.g. frequency (requests per time interval). The purpose in throttling is to ensure a proper quality of service (QoS) for every client and also block potential data-mining bots.

Cows Gateway offers a realtime dashboard for monitoring purpose. This Cows Admin UI offers analytic visualizing capabilities for the administrators and according to this data seen on dashboard the admin can change the policy values if needed. Cows Gateway logs events in the background and from the Cows Admin UI facade the administrators can print out different kind of reports even from a long-term perioids.




Product feature categories

## Organization and management

Project team consisted of two project leaders Tommi Kivistö and Timo Virtanen. Project staff originally consisted of five people. Team members were Heikki Hämälistö, Joni Karvinen, Muhammad Faisal, Navaneetha Kotha and Zhang Zhang. Tommi Kivistö and Timo Virtanen were responsible the project management and documentation. Joni Karvinen and Heikki Hämälistö were responsible of User interface, including UI design, coding and test.Navaneetha Kotha, Muhammad Faisal and Zhang Zhang were responsible of background programming. Muhammad quitted the project.


From left to right: Zhang, Heikki, Tommi, Joni and Navaneetha. Timo is a team spirit. Muhammad is missing from the picture.

## Methods and tools

Cows Gateway, is a common gateway for web services not related to a certain service or content type. This Cows works independently in a Tomcat container on a web

server using document oriented MongoDB as its database. Software is platform independent. Back-end is written in Java and facade front-end is www-based created using Play! Framework and different Javascript libraries e. g. Highcharts. Morphia libraries in back-end and Play! module in front-end was used for Object Relational Mapping (ORM). Back-end used RESTlet framework for HTTP request handling.

Eclipse was mostly used as IDE and partly Netbeans.

Redmine worked as our project management and documentation tool. Calendar and gantt-charts were rarely used after few first months. Writing documents on Redmine Wiki wasn't collaborative and the writing itself to Wiki proved to be painfully ineffective. The final documentation was therefore copy-pasted away from Redmine and finalized using Google Docs. Flowcharts were drawn using Google Draw. Redmine was used for logging time and it is a good tool for that.

First IRC was intented as primary communication method but it didn't work out because only PMs had earlier experience using IRC. PMs used IRC themself for communication. Neither did Redmine Forums worked out so e-mail was mostly used later on for communication.

SVN worked out as our version control system. Also files were shared through it but also Dropbox and Redmine Files were used.

## Project phases and development model

Incremental software development model were indented to be used but following it failed badly and the model wasn't applied in any way. We thought to have separate increments for the Cows Gateway back-end and Cows Admin UI front-end facade. Due to difficulty in defining the requirements, functions and data structure we couldn't clearly form suitable increments but more the idea of the construction of our application changed every week. Making decisions was hard and this ended up that we had only a one large increment as a total.

Forming a coherent vision of our project goals was difficult and the starting phase took much time. Starting phase problems were late receive of the project, communication (use of IRC), basic tool problems (using Redmine) and getting known to new technologies (Mongo & REST). The first half of the project was mostly random inefficient undetermined "getting known to techniques" and "forming the coherent vision of our goals" kind of things and this "starting" phase took approximately half of our project time. We couldn't complete this defining the vision, functions, data and requirements but only in the later half of our schedule (around New Year 2012). First half might be described as a sort of chaos where the direction was lost and never completely retrieved.

Project goals and technologies came clear during the early 2012 and a serious work started them but this was far too late to complete the project successfully. The challenge was just too big for the resources that we had. We ended up having a complete documentation (requirements, functions and data) but only a half-working software. Our back-end is working but wasn't completed as a whole and so wasn't the UI. Also the integration of the back-end and front-end wasn't completed.

Despite of the many setbacks we may call the project successful. We had the product and documentation deliverable completed and all the team members had learned a lot during the project.

| What | Completed |
|---|---|
| Preliminary analysis | 23.9.2011 |
| Project plan | 7.10.2011 |
| Increments (1st) | 4.3.2012 |
| Final meeting with the customer, project cd | 15.3.2012 |

## Experiences

We used some new technology that our team members had never touched before, including MongoDB, Restlet, Morphia and Play! Framework. We spent a lot of time on studying and making little demos for these new technology. But we found that we were still not prepared enough when we try to implement and integrate the whole system. Learning so much in a relative small time period proved to be very challenging. We got to known that it's hard to design a software using some technology if you don't understand it well.

Thinking about the resources; one of our team member left the project. We had only two members for background programming and testing. They had to study and use new technology at the same time. But not only the studying things were a major challenge but defining the goals, requirements and functions proved to be very challenging when technologies are relative unknown. We also had some trouble with the requirement engineering. We changed our models several times because of the lack of user requirements and this was partly because lack of understanding the project scope.

We spent too much time on the management tools (Redmine) which were thought to be not so important as we expected before. Writing longer documents to wiki proved to be slow and we should have moved to using Google Docs much earlier. Wiki is suitable only for small documenting. Using version control (SVN) was also inefficient at first because most of team members hadn't used it before.

After all, project was still considered to be successful in spite of these negative lessons, we finished the project and learned real developing experience about some new technology. And we also got precious experience about management, documentation and team work.

Time management and enough time is absolutely crucial for the succession. Team members had some dedication problems due to taken too many other courses on the same time and project work is definitely a course that takes a lot of time and effort.

## Statistics

| Team size | Dev. model | Start date | End data | Days | Hours |
|---|---|---|---|---|---|
| 2+4*(+1 quit)* | Incremental | 16.9.2011 | 15.3.2012 | 181 | 1225 |

Table 1: General project information.

| Acti-vity | Planning and mana-gement | Req. specifi-cation. | De-sign | Code | Integ-ration and testin g | Reviews | Repai r | Study | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours | 451.65 | 33.75 | 40.05 | 308.5 | 0 | 18.2 | 0 | 355 | 17.5 | **1225** |
| % | ~37% | 3% | ~3% | ~25% | 0% | 1.5% | 0% | 29% | 1.5% | 100% |

Table 2: Group effort by activity.

| Document | Pages | Versions |
|---|---|---|
| Preliminary analysis | 3 | 1 |
| Project Plan | 10 | 4 |
| Usability analysis | N/A | N/A |
| Requirements specification | 10 | 3 |
| Design plan | 15 | 6 |
| User interface document | N/A | N/A |
| Test plan | N/A | N/A |
| Test report | N/A | N/A |
| Usability test report | N/A | N/A |
| Final report | 14 | 2 |
| Project's story | 6 | 2 |

Table 3: Documents.

| Language | JAVA |
|---|---|
| LOC | 692 |
| SLOC | 437 |
| Reused code | Morphia, RESTlet |
| Reused and modified | 0 |

Table 4: Cows Gateway (Back-end) codelines. Doesn't include Morphia and RESTlet frameworks.

| Language | HTML/CSS/Javascript/Play |
|---|---|
| LOC | 2786 |
| SLOC | 2235 |
| Reused code | jQuery, Highcharts, Play! |
| Reused and modified | 0 |
| Source files | 31 |

Table 5: Cows Admin UI (front-end) codelines. Doesn't include jQuery, Highcharts and Play! Framework.